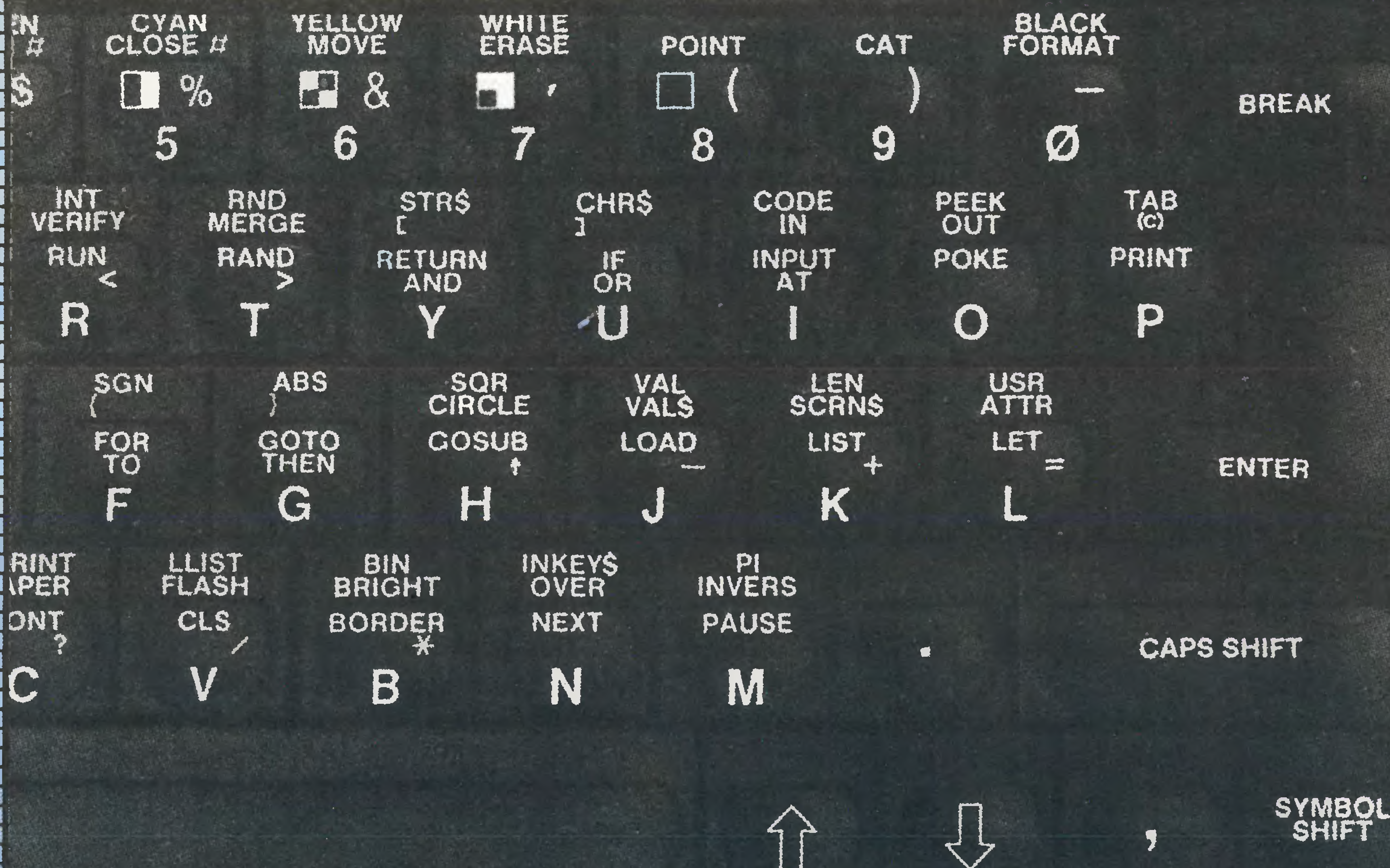
abc

!KOMPUTER

Andrzej Kadlof

Tajniki ZX Spectrum



Spis treści

Wstęp

Mikrokomputer ZX Spectrum firmy Sincair Research dzięki
*niskiej cenie
*stosunkowo bogatej, jak na produkt z 1982 r., grafice
*bogactwu łatwo dostępnych programów
*prostocie obsługi i programowania
*jawności i dostępności szczegółów budowy komputera i oprogramowania systemowego

zdobył na polskim rynku dominującą pozycję. Jego następcami są zgodne z nim programowo komputery Timex 2048 sprzedawane w Baltonie i Centralnej Składnicy Harcerskiej, a modele Timex 2068 (Unipolbrit 2086), ZX Spectrum 128k, ZX Spectrum 2+ (najnowsza propozycja Amstrada—Sinclaira) i Elwro 800 Junior — przyszły polski komputer edukacyjny — mogą w jednym z trybów swej pracy naśladować ZX Spectrum.

Broszura ta pomyślana jest jako uzupełnienie firmowej instrukcji obsługi. Dla użytkowników nie mających dostępu do oryginalnej dokumentacji lub ze względów językowych nie będących w stanie z niej skorzystać przytaczamy także skrót najważniejszych informacji zawartych w instrukcji. Z żalem musieliśmy natomiast do innej okazji odłożyć omówienie dodatkowych możliwości ZX Spectrum wyposażonego w ZX Interface 1 (obsługa ZX Microdrive, RS 232 i sieci lokalnej).

Lakoniczność przykładów zmusi wielu Czytelników do eksperymentalnego wyjaśnienia wątpliwości. Jest to najlepsza droga poznania własnego komputera — naciskając klawisze nie sposób go zepsuć!

Broszura ta nie jest przeznaczona do systematycznego studiowania. Dobór i rozkład materiału powinny ułatwić odszukanie potrzebnych przy pracy z komputerem informacji i rozstrzygnięcie wątpliwości zarówno początkującym, jak i zaawansowanym użytkownikom ZX Spectrum, pozwolić im lepiej zrozumieć jego działanie.

KLAWIATURA

Klawiatura ZX Spectrum składa się z 40, a ZX Spectrum + z 58 klawiszy. Słowa kluczowe języka Basic wprowadza się jednym naciśnięciem klawisza, a nie — jak w innych komputerach — wpisując poszczególne litery. Wprowadzane znaki i słowa wyświetlane są na dole ekranu, w miejscu wskazywanym przez migający prostokąt zwany **kursorem** (nie jest on widoczny bezpośrednio po uruchomieniu urządzenia lub pojawieniu się komunikatu, ukazuje się dopiero po naciśnięciu klawisza!). Litera wewnątrz kursora wskazuje, czego komputer się spodziewa, czyli w jakim znajduje się trybie (ang. mode) wprowadzania danych i jak zrozumie naciśnięcie kolejnego klawisza:

K — słowa kluczowe (KEYWORDS)
L — małe litery (LETTERS)
C — DUŻE LITERY i niektóre symbole (CAPITALS)
E — słowa kluczowe i znaki umieszczone pod i nad klawiszami (tryb rozszerzony, EXTENDED MODE)
G — symbole graficzne (tryb graficzny, GRAPHICS)

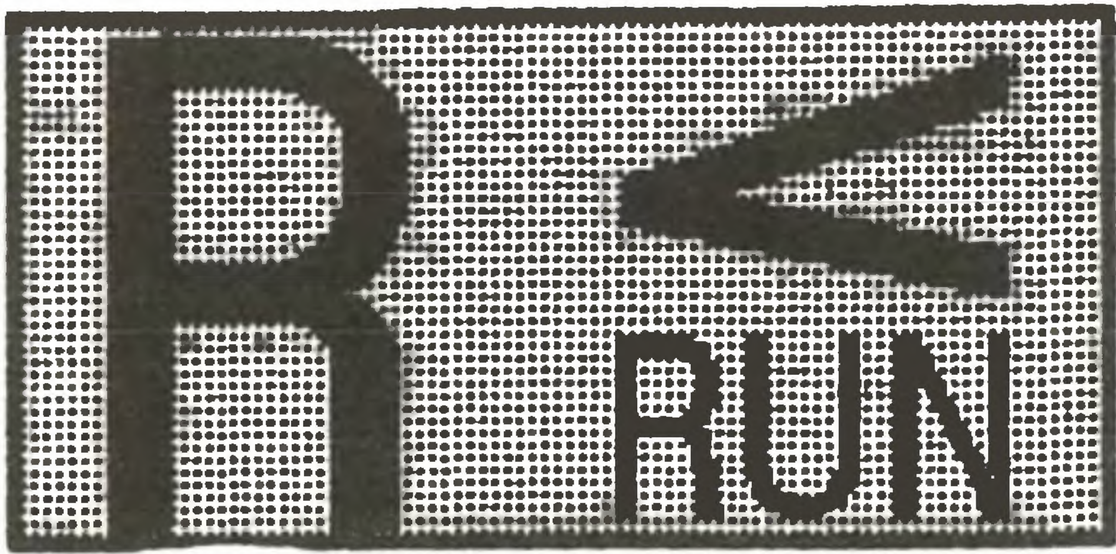
Znaczenie klawisza zależy — oprócz stanu kursora — także od wcześniejszego wciśnięcia (lub nie!) i równoczesnego trzymania jednego z dwóch klawiszy specjalnych:

CAPS SHIFT — DUŻA LITERA (Capitals shift, w skrócie CS) i
SYMBOL SHIFT — symbol (w skrócie SS)

Korzystając z nich należy najpierw wcisnąć CAPS SHIFT (lub SYMBOL SHIFT), a następnie, bez puszczania klawisza funkcyjnego, właściwy klawisz, np. cyfrę 2. Postępowanie takie oznaczać będziemy symbolem "/„ , np. SS/2.

Jeden klawisz może mieć w ZX Spectrum nawet osiem różnych znaczeń. Oto, co zostanie wyświetlone zależnie od stanu kursora i wciśniętych klawiszy specjalnych po naciśnięciu dowolnego klawisza z literą (np. R):

INT



tryb E: **INT**

tryb L: r tryb C: **R**	tryb L,C lub K + SS/: znak <
tryb K: **RUN**	

tryb E + SS/: **VERIFY**

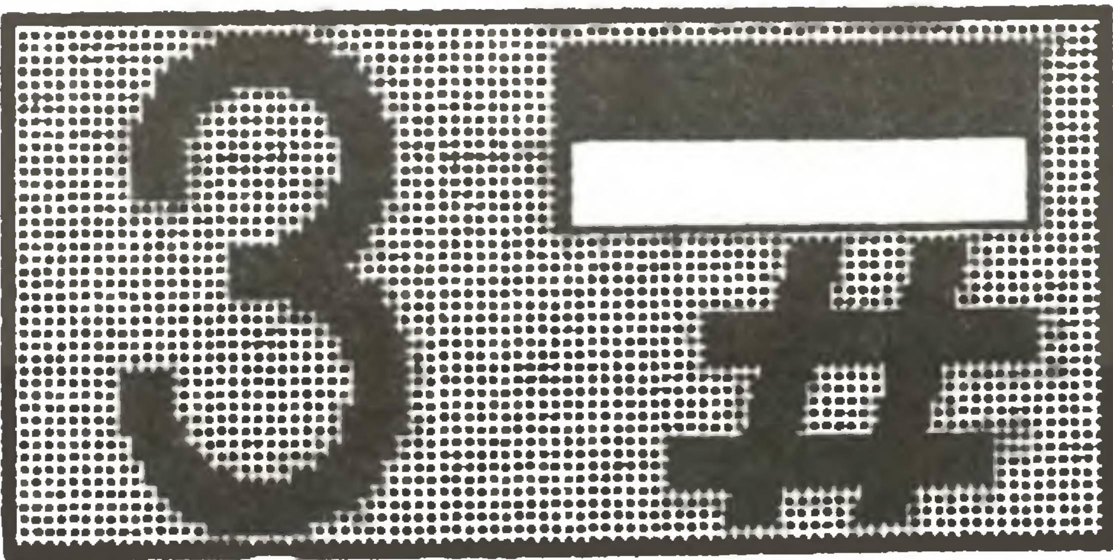
tryb G: **znak w kształcie litery R lub innym, zdefiniowanym przez użytkownika** (dla klawiszy A—U); klawisze V,W,X,Y,Z są tu bezużyteczne

układ rysunku zgodny jest z opisem klawisza na klawiaturze wersji podstawowej ZX Spectrum

Tryb C dla tej grupy klawiszy jest równoważny trybowi L z jednoczesnym stałym wciskaniem CS i oznacza wyświetlanie DUŻYCH LITER.

Nad niektórymi klawiszami górnego rzędu, np. nad klawiszem z cyfrą 3, umieszczono po dwa napisy:

MAGENTA



tryb E + CS/: **MAGENTA (kolor znaku)**
tryb E: **MAGENTA (kolor tła)**
tryb K,L lub C + CS/: **TRUE VIDEO**

Tryb K,L lub C cyfra 3	tryb G + CS/: ▬ tryb G : ▬
	tryb K,L lub C + SS/: znak #

tryb E + SS/: **LINE**

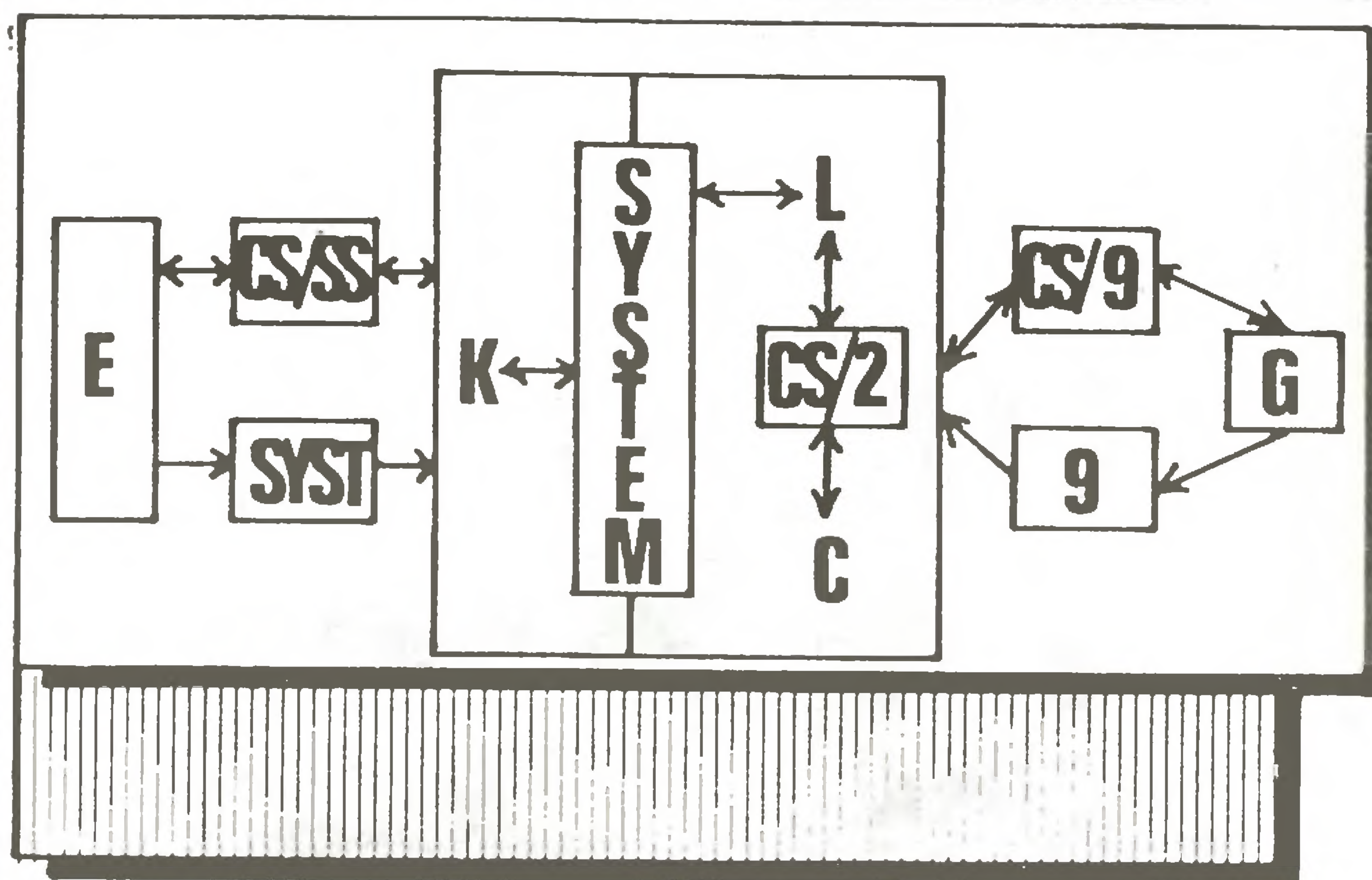
W trybie G po wciśnięciu takiego klawisza na ekranie pojawi się narysowany na nim symbol graficzny, a w wypadku równoczesnego wciśnięcia CS lub SS — ten sam symbol z odwróconymi kolorami. Napis nad klawiszem oznacza symbole kontrolne. Nie są one wyświetlane jako znaki na ekranie, a jedynie wywołują pewne efekty, np. w trybie rozszerzonym klawisz taki określa kolory tła i atramentu następnych znaków:

tryb E + CS/ (kolor znaku):		tryb E (kolor tła):
CS/0	czarny(BLACK)	0
CS/1	ciemnoniebieski(BLUE)	1
CS/2	czerwony(RED)	2
CS/3	fioletowy(MAGENTA)	3
CS/4	zielony(GREEN)	4
CS/5	jasnoniebieski(CYAN)	5
CS/6	żółty(YELLOW)	6
CS/7	biały(WHITE)	7
CS/8	równoważne FLASH 1	
	równoważne BRIGHT 1	8
CS/9	równoważne FLASH 0	
	równoważne BRIGHT 0	9

Zauważmy, że klawisze 8 i 9 również mają w tym trybie znaczenie mimo braku odpowiedniego oznaczenia na klawiaturze. Przy kursorze K,L lub C jedynie kombinacje CS/3 i CS/4 wywołują efekty barwne:

CS/4 powoduje zamianę kolorów tła i atramentu następnych znaków (INVERSE VIDEO), a
CS/3 przywraca ich pierwotny układ (TRUE VIDEO; w ZX Spectrum+ kombinacjom tym odpowiadają osobne klawisze).

Zwróćmy uwagę, że posługiwanie się powyższymi kombinacjami klawiszy jest jedynym sposobem zmiany kolorów tekstu (listy rozkazów) programu. Pozostałe kombinacje omawiamy dalej (w rozdziale o edytorze), teraz zajmijmy się sposobami zmiany stanu kursora. Na rysunku słowo SYSTEM sygnalizuje, że o zmianie trybu decyduje sam komputer:



Jak widać nie można przejść wprost z trybu G do E. Użytkownik nie może również wpływać bezpośrednio na przechodzenie komputera z trybu K w L i C oraz na odwrót. System ten jest skomplikowany tylko na pierwszy rzut oka: już po kilku próbach wszystko okazuje się proste i zaczniemy nawet dostrzegać pewne zalety takiego rozwiązania klawiatury.

Klawisz **BREAK/SPACE** w czasie wprowadzania danych służy do wprowadzania spacji, czyli odstępów między znakami. Znaczenie słowa BREAK i klawisza ENTER omawiamy w następnym rozdziale.

W modelu ZX Spectrum+ klawiatura jest bardziej rozbudowana i wygodniejsza w użyciu. Zmiany trybów dokonuje się pojedynczymi, dobrze oznakowanymi przyciskami. Również wszystkie symbole kontrolne uzyskiwane w klasycznym modelu przez CS/cyfra tu dostępne są za pomocą oddzielnych klawiszy, wydzielono także najczęściej stosowane znaki przestankowe. Informacje pisane nad i pod klawiszem tu znalazły się na górnej jego części. Pozostałe zasady wykorzystywania i zmiany trybów są identyczne.

Rozdział II

EDYTOR

Edytor to zawarty w pamięci stałej (ROM) program pozwalający na komunikowanie się człowieka z komputerem: wprowadzanie danych, ich modyfikowanie i poprawianie. Wszystko, co piszemy na klawiaturze, pojawia się zazwyczaj w dolnej części ekranu. Każdy tekst w tym obszarze można modyfikować, czyli "edytować". Klawisze CS/5 i CS/8 (strzałki w lewo i prawo w Spectrum+) pozwalają przesuwać kursor w lewo i prawo wzdłuż edytowanej linii bez wymazywania jakichkolwiek znaków. Do kasowania symbolu służy klawisz CS/0 (DELETE w Spectrum+). Wymazywany jest znak położony bezpośrednio przed kursorem, przy czym słowa kluczowe języka Basic traktowane są jako pojedyncze znaki.

Podczas pisania programu lub poleceń do natychmiastowego wykonania edytor w wielu wypadkach (po wprowadzeniu słowa kluczowego, dwukropka, THEN itp.) sam zmienia tryb wprowadzania (między K i L lub C), chroniąc użytkownika przed przypadkowymi błędami. Jeśli komuś na tym szczególnie zależy można go "oszukać" wstawiając np. dwukropek dla uzyskania trybu K i następnie kasując go. Oszustwo takie oczywiście nie usuwa późniejszych ewentualnych kłopotów z wprowadzeniem błędnej składniowo linii do programu.

Koniec edycji, wpisywania ciągu rozkazów lub kolejnej linii programu sygnalizujemy klawiszem ENTER. Komputer błyskawicznie sprawdza poprawność syntaktyczną (tj. zgodność z zasadami składni języka Basic) tekstu i w razie potrzeby w niezrozumiałym miejscu pojawia się migający znak zapytania. Może on wystąpić znacznie dalej niż rzeczywisty błąd, np. przy braku jednego z nawiasów zamykających w skomplikowanym wyrażeniu Spectrum nie jest w stanie zgadnąć, gdzie należałoby go dostawić i błąd zasygnalizuje dopiero na końcu wyrażenia. Zaakceptowanie linii przez komputer nie gwarantuje jednak jej poprawności logicznej ani zgodności z intencjami autora!

Nie ma możliwości zmuszenia komputera do zapamiętania błędnej (w sensie składni języka Basic, a nie logiki programu) sekwencji rozkazów. Oznacza to m.in., że bez dołączonego ZX Interface 1 niemożliwe jest nawet przygotowanie programu sterującego współpracą z pamięcią typu Microdrive, gdyż nie będzie można wprowadzić specyficznych dla tego urządzenia instrukcji.

Jeżeli w poprawianym wierszu umieszczono symbole kontrolne przesuwany kursor zachowuje się czasem dziwacznie i nietypowo (chowa się, cofa itp.). Nie należy się tym przejmować. W trakcie kasowania takich symboli mogą pojawić się znaki zapytania, które należy kasować oddzielnie — symbole te zajmują zazwyczaj w pamięci dwa bajty i Spectrum zmuszone do wyświetlania znaku o kodzie mniejszym od 32 pisze w jego miejsce znak zapytania, jeśli nie potrafi go inaczej zinterpretować.

Aby poprawić już wprowadzoną do pamięci linię programu trzeba ją najpierw skopiować do dolnej części ekranu. W tym celu przesuwamy wskaźnik linii bieżącej (strzałka > za jej numerem) w górę i w dół klawiszami CS/6 i CS/7 (strzałki w górę i w dół w Spectrum+). Przy dłuższych programach szybsze bywa posłużenie się instrukcją LIST n. Można również wprowadzić liczbę nieco mniejszą od numeru potrzebnej linii, jeśli jesteśmy pewni, że nie ma w programie linii o tym numerze, którą moglibyśmy niechcący skasować. Po naciśnięciu ENTER wskaźnik będzie ustawiony na pierwszej linii o numerze większym od podanego lub ostatnią, jeśli takiej nie ma. Wskaźnik > po takim zabiegu może być chwilowo niewidoczny na ekranie, ale po naciśnięciu CS/1 (EDIT w Spectrum+) żądana linia zostanie skopiowana do obszaru roboczego edytora.

Wciśnięcie samego ENTER przy pustej dolnej części ekranu powoduje wyświetlenie fragmentu programu w okolicach linii bieżącej (tzn. wskazywanej przez strzałkę >). Jednorazowo można wprowadzić wiersz o dowolnej długości. Początkowo dolna część ekranu będzie się automatycznie rozszerzała, lecz stopniowo komputer będzie na wciskane klawisze reagował coraz wolniej, sygnalizując rosnącą dezaprobatę dla działań człowieka. Po zapełnieniu 22 linii protest komputera stanie się znacznie gwałtowniejszy. Przestanie wyświetlać dalsze znaki wprowadzane z klawiatury i na każde naciśnięcie klawisza zareaguje przykrym buczeniem, lecz mimo to nadal przyjmie i zapamięta wszystko to, co zechcemy napisać. Jego opory można złagodzić modyfikując odpowiednie zmienne systemowe. Klawisz CS/1 (EDIT) pozwala również na jednorazowe kasowanie całej dolnej części ekranu.

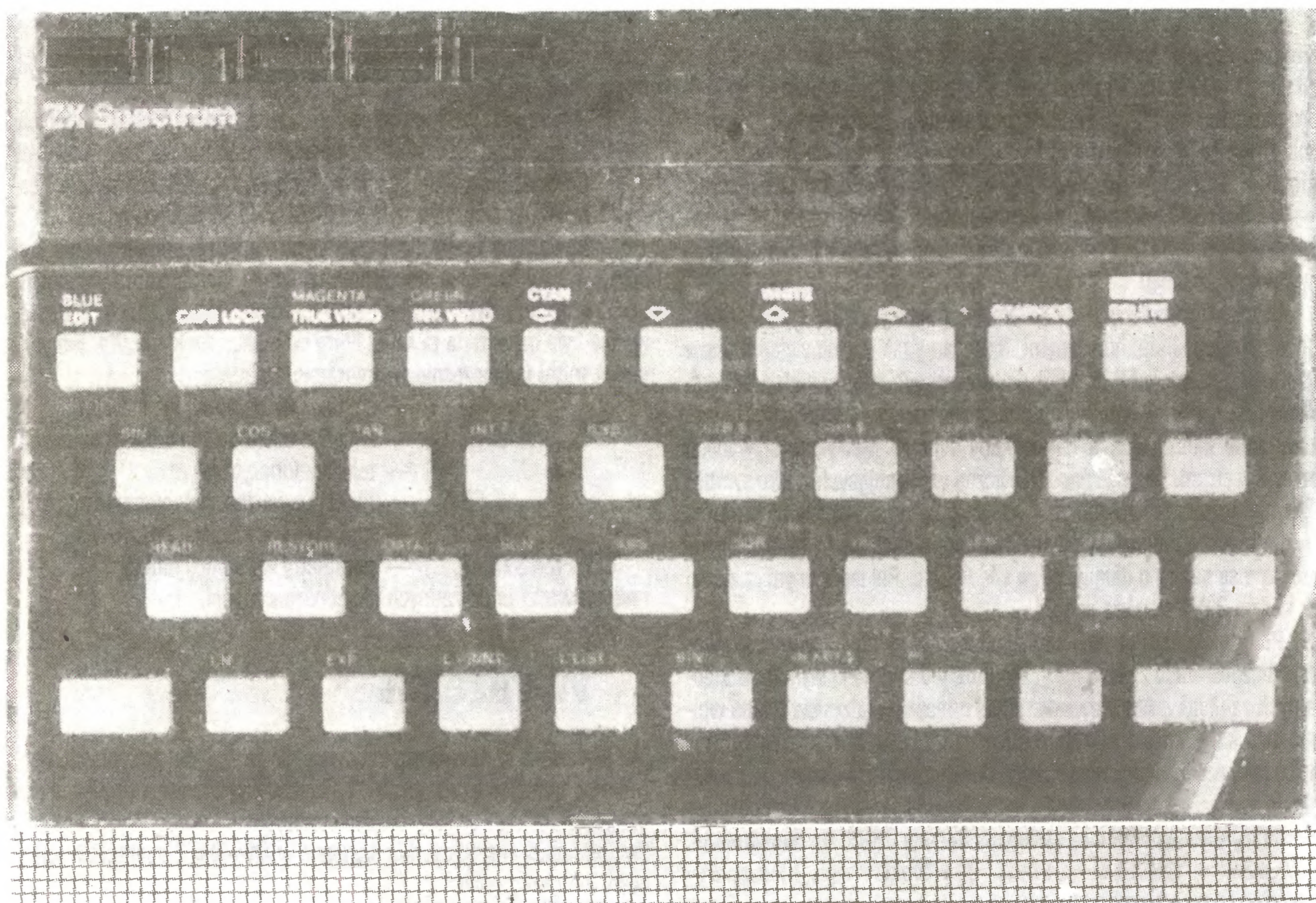
Rozdział III

BASIC

W rozdziale tym przedstawimy skrótowo alfabetyczny wykaz funkcji i rozkazów języka ZX—Basic. Warto go przejrzeć nawet dobrze znając ten dialekt, gdyż podajemy też informacje pominięte w oryginalnej instrukcji obsługi.

W wykazie posługujemy się następującymi oznaczeniami:

@ — pojedyncza litera
v — zmienna dowolnego typu
x, y, z — wyrażenie numeryczne o wartościach rzeczywistych
k, m, n — wyrażenie numeryczne o wartości automatycznie przybliżanej przez najbliższą liczbę całkowitą
e — wyrażenie numeryczne lub napis (łańcuch)
f — napis (łańcuch, ang. STRING)
s — ciąg rozkazów oddzielanych od siebie dwukropkiem
c — ciąg instrukcji określających kolory i sposób wyświetlania (INK, PAPER, BRIGHT, FLASH, OVER, INVERSE) oddzielanych średnikami lub przecinkami



Symbol umieszczony w [nawiasie kwadratowym] można pominąć, np. LOAD f CODE [m[,n]] oznacza, że poprawne są formy

LOAD f CODE

LOAD f CODE m

oraz

LOAD f CODE m,n

W takim wypadku Spectrum zazwyczaj uzupełnia brakujące parametry wartościami domyślnymi wynikającymi z kontekstu, w naszym przykładzie — odczytanymi z taśmy. Zapis @[$] oznacza z kolei @$ lub @, czyli dopuszcza użycie zarówno zmiennej numerycznej, jak i tekstowej.

Spectrum niemal we wszystkich sytuacjach dopuszcza użycie jako parametrów wyrażeń wymagających uprzedniego obliczenia, ograniczeniom podlegają jedynie dopuszczalne wartości obliczanych wyników i ich typ. Wyrażenia typu k, m, n mogą dawać w wyniku dowolne liczby rzeczywiste o wartości bezwzględnej mniejszej od 65535 (2 ↑ 16-1), które zostaną przybliżone przez najbliższe liczby całkowite. Pojawienie się zbyt dużej liczby sygnalizowane jest jako błąd komunikatem B (Integer out of range).

Struktura programu

Program w języku Basic to ciąg instrukcji podzielony na linie. Każda linia zaczyna się od swego numeru, czyli liczby naturalnej z zakresu 1...9999. Numerowane wiersze mogą być wprowadzane w dowolnej kolejności, zostaną one automatycznie uporządkowane wg rosnących numerów. Nie jest konieczne numerowanie kolejnymi liczbami, np. 1,2,3..., taki styl programowania jest wręcz odradzany. Wygodniej jest posługiwać się numerami 10,20,30..., co ułatwia późniejsze wstawianie nowych linii między już istniejące.

Wprowadzenie nowej linii o numerze już istniejącym powoduje zastąpienie starej wersji przez nową, a sekwencja klawiszy NUMER—ENTER usuwa z programu wiersz oznaczony daną liczbą.

Dopuszczalne jest umieszczanie w tekście linii zawierających, oprócz numeru, jedynie spacje (przynajmniej jedną) celem optycznego wydzielania w wydruku wybranych bloków programu.

Pojedynczy wiersz programu może być znacznie dłuższy od jednej linii ekranowej (32 znaki), nie może jednak zawierać więcej niż 127 instrukcji. Rozkazy w wierszu muszą być oddzielane dwukropkiem. Pomiędzy kolejnymi instrukcjami w linii można umieszczać dowolną liczbę spacji celem uzyskania "wcięć" podnoszących czytelność wydruku.

Wciśnięcie klawisza ENTER kończy wprowadzanie linii. Komputer sprawdza jej zgodność z regułami języka (patrz rozdz. Edytor) i jeżeli rozpoczynała się ona od numeru dołącza ją do istniejącego programu wstawiając na właściwe miejsce, a gdy numer pominięto przystępuje natychmiast do wykonania podanego ciągu rozkazów.

Typy danych

ZX—Basic pozwala bezpośrednio operować na liczbach zmiennoprzecinkowych i na tekstach. Liczby można zapisywać w jednej z trzech postaci:

— **całkowitej**, czyli jako ciąg cyfr, ew. poprzedzonych znakiem + lub —, np. 0, +17, —337498;

— **ułamkowej**, czyli jako dwa ciągi liczb oddzielone kropką (jeden z nich może być pusty). Całość można poprzedzić znakiem + lub —, np. +31.1769.3330.12—.72 itp ;

— **wykładniczej**, czyli jako liczby w postaci całkowitej lub ułamkowej uzupełnionej o wykładnik dziesiętny, będący liczbą całkowitą. Wykładnik dziesiętny podaje do jakiej potęgi trzeba podnieść 10, by po pomnożeniu wyniku przez pierwszą część zapisu liczby uzyskać jej wartość. Wykładnik ten oznaczamy literą e lub E, np. zapis 7.32E—7 oznacza liczbę $7.32*10^{-7}$ czyli 0.000000732.

Bezwzględna wartość liczby w ZX Spectrum nie może przekraczać 1.7E38. Liczby mniejsze od 10^{-39} będą uznane za równe zeru. Pamiętać należy, że liczby są przechowywane z dokładnością do jedynie 9—10 cyfr znaczących, np. 5E33 oraz 5E33 +2 to dla komputera ta sama liczba, gdyż ich wartości dokładne różnią się dopiero na dalszych miejscach. Spacje wewnątrz zapisu liczby są niedopuszczalne. Spectrum do wyświetlenia liczby potrzebuje co najwyżej 14 znaków. Wartości wymagające nie więcej niż 8 cyfr są podawane dokładnie, pozostałe zaś w postaci wykładniczej.

W większości odmian języka Basic liczby całkowite i zmiennoprzecinkowe reprezentowane są w pamięci komputera jednolicie. W dialekcie ZX—Basic odstąpiono od tej reguły nie pozostawiając jednak programistom wpływu na sposób reprezentacji poszczególnych wartości. **Choć w Spectrum wszystkie liczby są przechowywane w pięciu bajtach, to wielkości całkowite z przedziału: —65536...65535 mają odmienną postać** (patrz rozdział "Arytmetyka komputerowa"). Operacje arytmetyczne na takich liczbach wykonywane są szybciej!

Drugim typem danych w ZX—Basic są napisy, zwane też łańcuchami lub stałymi alfanumerycznymi. Nazywamy tak dowolny ciąg znaków (symboli o kodzie od 0 do 255) umieszczonych między cudzysłowami. Wewnątrz napisu można więc umieszczać rownież symbole kontrolne, kody słów kluczowych a nawet kody symboli niewykorzystywanych przez Spectrum. Długość napisu to liczba występujących w nim symboli, a nie liczba znaków potrzebnych do jego wyświetlenia na ekranie. Liczba ta jest ograniczona jedynie rozmiarem dostępnej pamięci. Dopuszczalne są napisy o długości 0, a więc puste. Pełnoprawnym znakiem (o kodzie 32) jest także spacja.

Podczas konstruowania napisu większość symboli można wprowadzać bezpośrednio z klawiatury. Niektóre jednak mogą wymagać specjalnych zabiegów. Aby wewnątrz stałej umieścić cudzysłów trzeba wpisać go z klawiatury dwukrotnie (pojedynczy cudzysłów interpretowany byłby jako oznaczenie końca stałej). Uniwersalnym sposobem wstawiania do stałej tekstowej symboli o kodach niedrukowalnych jest posłużenie się wyrażeniem alfanumerycznym i funkcją CHR$. Np. konstrukcja: "TO JEST" + CHR$ 13 + "NAPIS" da w efekcie napis, którego druga część wyświetlana będzie zawsze w osobnym wierszu:

TO JEST
NAPIS

ZX—Basic nie zna stałych logicznych. Tę rolę pełnią liczby. Wartości różne od zera są interpretowane jako prawda, równe zaś zeru jako fałsz. Wynikiem obliczenia wartości wyrażeń logicznych są liczby 0 lub 1.

Obok stałych ZX—Basic dopuszcza stosowanie zmiennych numerycznych i tekstowych. Służą one do przechowywania w pamięci komputera stałych odpowiedniego typu. Zmienne są identyfikowane przez swoje nazwy, które nadaje im programista. Nazwom w ZX—Basic poświęcamy następny punkt tego rozdziału. Możliwość stosowania nazw zwalnia programistę od konieczności pamiętania gdzie i w jakiej postaci są przechowywane poszczególne wartości.

Zmienne przechowujące pojedynczą liczbę lub łańcuch znaków nazywamy zmiennymi prostymi. Oprócz nich możemy również stosować tzw. zmienne tablicowe (tablice), muszą one jednak być zadeklarowane instrukcją DIM. Dopuszczalne są zarówno tablice numeryczne jak i znakowe. Elementem tablicy numerycznej jest liczba, a znakowej — pojedynczy symbol o kodzie od 0 do 255. Wiersze tablic znakowych mogą być używane jako zmienne alfanumeryczne o stałej długości określonej przez ostatni indeks (wymiar) deklaracji danej tablicy. Liczba wymiarów jest ograniczona do 255. Poszczególne indeksy nie mogą przekraczać wartości 65535. W praktyce wielkości te muszą być znacznie mniejsze ze względu na dostępną pamięć. W deklaracji tablicy podaje się jedynie górne ograniczenia indeksu. Jako ograniczenie dolne zawsze jest przyjmowana wartość 1.

Nazwy

Jedną z wad ZX—Basic są poważne ograniczenia w swobodzie definiowania nazw używanych zmiennych. Jedynie proste zmienne numeryczne stwarzają nam pole do popisu. Nazwą takiej zmiennej może być dowolny ciąg złożony z liter, cyfr i spacji. Pierwszym symbolem nazwy musi być litera.

Wszystkie inne zmienne muszą mieć nazwy jednoliterowe, w tym również zmienne sterujące pętlami FOR...NEXT.

Zmienne tekstowe różnią się od numerycznych znakiem $ umieszczonym po literze. Dotyczy to również nazw funkcji definiowanych przez użytkownika dyrektywą DEF FN oraz ich argumentów.

Do tworzenia nazw w programie można używać zarówno małych, jak i dużych liter, Spectrum automatycznie zamienia duże litery na małe. Również wszystkie spacje są ignorowane. Dla komputera nazwy:'TO JEST zmienna' oraz 'tojestZMIENNA' są identyczne.

Ograniczenie liczby dopuszczalnych nazw rekompensowane jest częściowo możliwością stosowania tej samej litery jako nazwy zmiennych różnych typów. Jedynym zastrzeżeniem jest, by zmienna tekstowa nie była oznaczana tą samą literą co tablica znakowa. Np. jedna litera A może jednocześnie w programie oznaczać:

A — zmienną numeryczną lub sterującą
A$ lub A$(i,j) — zmienną tekstową lub tablicę znakową
A(i) — tablicę numeryczna
FN A... — funkcję o wartościach numerycznych
FN A$ (...) — funkcję o wartościach tekstowych
DEF FN A $ (A,A$,...) — parametry w definicji funkcji

i nie prowadzi to do żadnych niejednoznaczności.

Wyrażenia

Wyrażenia w ZX—Basic budowane są ze stałych, zmiennych, operatorów, funkcji oraz nawiasów. Kolejność wykonywania działań w wyrażeniu określają nawiasy okrągłe i priorytety poszczególnych operacji. Poniżej omawiamy wszystkie dostępne w ZX—Basic operacje w kolejności ich priorytetów.

Wydzielenie fragmentu napisu

Operacja wykonywana może być na łańcuchach, zmiennych tekstowych oraz tablicach znakowych. Dla łańcuchów i zmiennych parametry określające wycinany fragment muszą być podane w nawiasach okrągłych, dla tablic dane te umieszcza się w polu ostatniego indeksu lub w razie jego pominięcia w osobnych nawiasach, analogicznie jak dla zmiennych prostych. Parametry te mogą mieć postać:

— puste, ciąg nie ulega zmianie
(k) — wyrażenie arytmetyczne, wynikiem jest pojedynczy symbol występujący na k—tym miejscu napisu
(k TO m) — wyrażenia arytmetyczne połączone słowem kluczowym TO, wynikiem jest fragment napisu obejmujący symbole od k—tego do n—tego włącznie. Gdy k>m>0 wynikiem jest łańcuch pusty, a gdy napis jest krótszy od m sygnalizowany jest błąd, np. po wykonaniu programu:

```
10 LET a$="Napis"
20 DIM B$ (2,3)
30 LET b$ (1)="abc"
40 LET b$(2)="XYZ"
```

otrzymamy:

```
a$ = a$() = a$(TO) = A$(1 TO 5) = "Napis"
A$(1) = a$(TO 1) = a$(1 TO 1) = "N"
a$(3 TO 3) = A$(3) = a$(5—2) = "p"
A$(TO 2) = a$(1 TO 2) = "Na"
a$(3 TO) = a$(3 TO 5) = "pis"
a$(2 TO 4) = "api"
b$(1) = B$(1,) = b$(1)() = b$(1,TO) = b$(1)(TO) =
      = b$(1,1 TO 3) = B$(1)(1 TO 3) = "abc"
b$(2,2) = b$(2)(2) = b$(2,2 TO 2) = b$(2)(2 TO 2) = "Y"
```

W innych dialektach języka Basic podobną rolę pełnią funkcje:
LEFT(a$,n) = a$(TO n),
RIGHT(a$,n) = a$(n TO) ,
MID(a$,k,m) = a$(k TO m),
TL(a$) = a$(2 TO).

↑ (operator potęgowania)

Argumentami operacji potęgowania są liczby rzeczywiste. Podstawa zawsze musi być liczbą nieujemną. Warto pamiętać, że a ↑ b obliczane jest jako EXP (b*LN a). Zatem np. a ↑ 2 będzie obliczane dłużej niż a*a. Zaletą drugiego sposobu jest także dopuszczalność ujemnych wartości a.

— (minus negujący wartość liczby)

Argumentem tej operacji jest zawsze liczba, wynikiem ta sama liczba o przeciwnym znaku. Minus negujący należy odróżniać od oznaczanej tym samym symbolem operacji odejmowania. Wysoki priorytet tej operacji pozwala na pisanie a*—b, co będzie interpretowane jako a*(—b). W innych dialektach języka Basic umieszczanie obok siebie dwóch operatorów jest niedopuszczalne.

*,/ (mnożenie i dzielenie)

Mnożenie i dzielenie arytmetyczne wykonywane są w kolejności, w jakiej występują w wyrażeniu. Znaku mnożenia nie można pomijać (zapis 2x zamiast 2*x jest niedopuszczalny). Mnożenie jest wykonywane w komputerze szybciej niż dzielenie. Lepsze jest więc użycie a*.5 zamiast a/2. (Patrz rozdział "Błędy w systemie" dotyczący kłopotów z dzieleniem).

+,— (dodawanie i odejmowanie)

Dla argumentów numerycznych symbole te oznaczają dodawanie i odejmowanie liczb. Obie operacje mają ten sam priorytet i są wykonywane w kolejności występowania.

Plus zastosowany do napisów i zmiennych tekstowych daje napis powstały przez dopisanie na końcu pierwszego napisu wszystkich symboli drugiego. W odróżnieniu od dodawania liczb nie jest to operacja przemienna (na ogół a$+b$ nie równa się b$+a$)

=,<,>,<=,>=,<> (porównania)

Są to dwuargumentowe operacje porównania wielkości tego samego typu. Wszystkie one mają ten sam priorytet i oznaczają kolejno relacje: równe, mniejszy, większy, mniejszy lub równy, większy lub równy oraz różne. Wynikiem tej operacji jest liczba 1, jeśli relacja zachodzi, albo 0 jeśli nie zachodzi. Wyniki testów mogą być używane w wyrażeniach arytmetycznych jako zwykłe liczby.

Jako przykład rozważmy następujący problem: mamy w programie tablicę numeryczną B zadeklarowaną instrukcją DIM B(100). Należy zbadać ile elementów tej tablicy jest większych od 7. Najprościej można to uzyskać rozkazami:

```
LET X=0 : FOR I=1 TO 100: LET X=X+(B(I)>7): NEXT I
```

Po ich wykonaniu zmienna X będzie zawierała żądaną wielkość.

Te same operatory stosowane są do napisów i zmiennych tekstowych. O wyniku operacji porównania ciągów znaków decyduje porządek leksykograficzny, za najmniejszy uważa się napis pusty (nie zawierający ani jednego znaku). Porównując kolejne symbole szuka się pierwszego miejsca, na którym oba ciągi różnią się. Za mniejszy uznaje się ten napis, w którym pierwszy różny symbol ma mniejszy kod. Np. prawdziwe są poniższe relacje:

""<"TEKST"<"TEKst"<"tekst"<"tz"<"tz0"

NOT (negacja logiczna)

Operacja ta neguje logicznie wartość wyrażenia numerycznego będącego argumentem, traktowanego jak wielkość logiczna.

$$\text{NOT } x = \begin{cases} 0 \text{ jeśli } x <> 0 \\ 1 \text{ jeśli } x = 0 \end{cases}$$

Pamiętamy, że wyrażenia logiczne typu (a>b) OR (c=d) komputer traktuje jako wyrażenia numeryczne.

AND (dwuargumentowy iloczyn logiczny)

Jest to bardzo wygodna operacja o zastosowaniach daleko wykraczających poza obliczanie samych wyrażeń logicznych. Pierwszym argumentem operacji AND może być zarówno wyrażenie numeryczne i alfanumeryczne, drugim musi być wyrażenie numeryczne. Oto definicja iloczynu logicznego, tak jak jest on rozumiany przez ZX—Basic:

$$x \text{ AND } y = \begin{cases} x \text{ jeśli } y <> 0 \\ 0 \text{ jeśli } y = 0 \end{cases}$$

$$f \text{ AND } y = \begin{cases} f \text{ jeśli } y<>0 \\ \text{"" jeśli } y=0 \end{cases}$$

Definicja ta może wydawać się dziwna i niezbyt związana z tym, co przywykliśmy uważać za iloczyn logiczny. Zwróćmy jednak uwagę, że jeśli operator AND znajdzie się między dwoma wyrażeniami logicznymi, to wynikiem iloczynu będzie 1 wtedy i tylko wtedy, gdy oba te wyrażenia będą prawdziwe, a więc przyjmą wartość 1. Zalety takiego określenia operacji AND ujawniają się w poniższych przykładach. Przypuśćmy, że zmiennej MAKSIMUM chcemy nadać wartość będącą większą z liczb a i b. Okazuje się, że wystarczy do tego jedna instrukcja:

```
LET MAKSIMUM=(a AND a>b) + (b AND a<=b)
```

Nawiasy w tym wyrażeniu są konieczne ze względu na wysoki priorytet operacji +.

Jeszcze bardziej spektakularne rezultaty możemy uzyskać operując napisami. Poniższy program rozpoznaje płeć użytkownika na podstawie jego imienia:.

```
10 INPUT "Podaj swoje imię ";a$
20 PRINT "Dzień dobry Pan"+("i" AND a$ (LEN a$)="a")
+("u"AND a$ (LEN a$)<>"a")
```

Nie dysponując tego typu operacją AND zmuszeni bylibyśmy do stosowania, nieraz wielokrotnie, instrukcji IF .. THEN oraz GO TO.

OR (suma logiczna)

Podobnie jak poprzednio jest to operacja dwuargumentowa. Oba argumenty muszą być wyrażeniami numerycznymi. Działanie tej operacji jest następujące:

$$x \text{ OR } y = \begin{cases} 1 \text{ jeśli } y<>0 \\ x \text{ jeśli } y=0 \end{cases}$$

Zastosowania OR poza wyrażeniami logicznymi są znacznie mniejsze niż AND. Na przykład instrukcję obliczającą maksimum z dwóch liczb można zapisać także w postaci

```
LET MAKSIMUM = a*(0 OR a>b) + b*(0 OR a<= b)
```

Zachęcamy do posługiwania się nawiasami nawet wtedy, gdy ze względu na priorytety nie są one konieczne. Pozwala to uniknąć przykrych i na ogół trudnych do wykrycia błędów.

Funkcje

ZX—Basic zawiera zestaw standardowych funkcji do obliczeń matematycznych, operacji na ciągach znaków oraz do badania ekranu i klawiatury. Z wyjątkiem funkcji ATTR, POINT i SCREEN$ argumenty będące stałymi lub pojedynczymi zmiennymi (prostymi lub tablicowymi) nie muszą być zamykane w nawiasy.

ABS x wartość bezwzględna liczby.

ACS x arcus cosinus; gdy x nie należy do przedziału <—1,1> sygnalizowany jest błąd A (Invalid argument). Wartość funkcji wyrażana jest w radianach.

ASN x arcus sinus. Uwagi jak wyżej.

ATN x arcus tangens. Wartość wyrażana w radianach.

ATTR (k,m) ustawienie atrybutów w k—tym wierszu i m—tej kolumnie. Sposób kodowania atrybutów opisany jest w rozdziale "Wykorzystanie pamięci". Gdy parametry nie mieszczą się w przedziałach 0<=k<23 oraz 0<=m<=31 sygnalizowany jest błąd B (Integer out of range).

BIN przekształcenie liczby binarnej (dwójkowej) bez znaku o maksymalnie 16 cyfrach do postaci dziesiętnej (na ekranie) lub typowej dla ZX Spectrum (w pamięci). ZX Basic nie zna funkcji odwrotnej, przekształcającej liczbę do postaci dwójkowej.

CHR$ k zamiana liczby na odpowiadający jej znak ASCII lub sterujący. Gdy k nie mieści się w przedziale <0,255> sygnalizowany jest błąd B.

CODE f funkcja odwrotna do poprzedniej: argumentem jest dowolne wyrażenie alfanumeryczne, a wartością funkcji kod pierwszego symbolu tego wyrażenia lub 0, gdy argumentem jest łańcuch pusty.

COS x cosinus. Argument musi być wyrażony w radianach.

EXP x funkcja wykładnicza e^x.

IN k badanie stanu portu wejścia/wyjścia procesora. Odpowiednik dwóch instrukcji asemblera Z80: LD BC,k oraz IN A,(C). Wartością funkcji IN jest liczba całkowita z przedziału 0...255. Jest to jedyna instrukcja, której beztroskie stosowanie może spowodować, że Wasz sprawdzony program nie będzie działał na innych egzemplarzach ZX-Spectrum! Wiąże się to z wykorzystywaniem IN do czytania klawiatury. Każdy rząd klawiszy podzielony jest na dwie grupy po pięć przycisków i każda piątka dołączona jest do pewnego portu:

klawisz	nr portu dziesiętnie	nr portu heksadecymalnie
CS...V	65278	# FEFE
A...G	65022	# FDFE
Q...T	64510	# FBFE
1...5	63486	# F7FE
0...6	61438	# EFFE
P...Y	57342	# DFFE
ENTER...H	49150	# BFFE
SPACE...B	32766	# 7FFE

Pięć młodszych bitów w danym porcie sygnalizuje, które klawisze są wciśnięte (0— wciśnięty, 1— zwolniony). Kłopot pojawia się przy trzech najstarszych bitach. W wersji 1 (najstarszej) ZX Spectrum są one zawsze równe 1, a więc IN 64510 = 255 oznacza zwolnienie klawiszy między Q a T. W wersji 2 bit szósty jest równy zeru i w opisanej sytuacji IN jest równe 191. W wersji 3 szósty bit jest nieokreślony i konieczne są dodatkowe operacje np. LET p=IN k: LET p=p—32*INT(p/32). Czytanie klawiatury przez IN umożliwia rozpoznanie wielu równocześnie wciśniętych klawiszy.

**INKEY$** Czyta klawiaturę. Wartością funkcji jest znak klawisza wciśniętego w momencie wywołania. Klawiatura czytana jest w trybie L lub C. Jeśli wciśnięto kilka klawiszy lub nie wciśnięto żadnego wartością INKEY$ jest łańcuch pusty. Funkcja ta nie czeka na wciśnięcie klawisza!

INT x część całkowita liczby. Liczba rzeczywista x jest zaokrąglana do największej liczby całkowitej nie przekraczającej x, np. INT 3.1 = INT 3.9 = 3, INT —5.1 = —6

LEN f długość łańcucha znaków.

LN x logarytm naturalny. Gdy x<=0 sygnalizowany jest błąd A.

PEEK k zawartość komórki pamięci o adresie k.

PI wartość stałej π = 3.14159265...

POINT (k,m) bada stan punktu ekranu. Funkcja ta ma wartość 1, gdy punkt na ekranie o współrzędnych (k,m) ma kolor atramentu (jest narysowany) i 0 jeśli ma on kolor tła, wykrywa więc punkty narysowane także wtedy, gdy kolor tła pokrywa się z kolorem atramentu i nie są one widoczne. Współrzędne muszą zawierać się w przedziałach 0<=k<=255 i 0<=m<=175.

RND generator pseudolosowy. Kolejne wartości równe są SEED/65536, gdzie SEED jest dwubajtową zmienną systemową o adresie 23671=#5C76. Za każdym wywołaniem RND jest ona modyfikowana zgodnie z wzorem

SEED= ((SEED + 1)*75 mod 65537) —1.

W ten sposób generowany jest ciąg o okresie 65536 i rozkładzie jednostajnym. Zmiennej SEED można nadawać wartości instrukcją RANDOMIZE k.

SCREEN$ (k,m) rozpoznaje znak na ekranie. Funkcja ta na podstawie kształtu rozpoznaje symbol wyświetlony w polu na przecięciu k—tej i m—tej kolumny. Rozpoznawane są jedynie symbole



o kodach od 32 do 127. W razie nierozpoznania znaku wartością funkcji jest łańcuch pusty. Współrzędne muszą być z przedziałów 0<= k <=23, 0<= m <=31.

SGN x znak liczby. Wartościami tej funkcji są zależnie od znaku argumentu liczby —1,0 lub 1.

SIN x sinus. Argument musi być wyrażony w radianach.

SQR x pierwiastek kwadratowy. Argument musi być liczbą nieujemną. Wartość x<0 sygnalizowana jest komunikatem A.

STR$ x zamiana liczby na łańcuch znaków. Funkcja ta zamienia liczbę na ciąg znaków, który byłby jej przedstawieniem na ekranie.

TAN x tangens. Argument musi być wyrażony w radianach i różnić się od wielokrotności PI/2.

USR działanie tej funkcji zależy od typu argumentu. Wywołanie jej z argumentem numerycznym (USR k) uruchamia kod maszynowy od adresu k. Po jej wykonaniu i powrocie do sterowania przez Basic wartością funkcji staje się zawartość pary rejestrów BC. Drugą możliwością jest wywołanie USR f. Wartością wyrażenia tekstowego f musi być jedna litera (mała lub duża) między "a" i "u" włącznie. Wartością funkcji jest adres w obszarze UDG, pod którym jest przechowywanych osiem bajtów określających kształt symbolu graficznego zdefiniowanego przez użytkownika i otrzymywanego w trybie G przez naciśnięcie klawisza z odpowiednią literą.

VAL f oblicza wartość wyrażenia numerycznego podanego w postaci ciągu znaków. Bardzo ważna funkcja o różnorodnych zastosowaniach. Jeśli wyrażenie f nie przedstawia poprawnego (w języku Basic) wyrażenia numerycznego sygnalizowany jest błąd C. W trakcie obliczania wartości mogą też wystąpić inne błędy zależne od postaci wyrażenia.

VAL$ f oblicza wartość wyrażenia tekstowego podanego w postaci ciągu znaków. Funkcja znacznie mniej przydatna od poprzedniej. Trudno wskazać zastosowania, w których użycie jej jest niezbędne, poza wyznaczaniem treści zmiennych tekstowych o nazwach nieznanych zawczasu i obliczanych dopiero w trakcie wykonywania programu. Przypuśćmy, że wartością zmiennej a$ jest litera określająca nazwę zmiennej alfanumerycznej. Nie dysponując funkcją VAL$ bylibyśmy w poważnym kłopocie chcąc np. wydrukować wartość tej zmiennej. Na szczęście mamy ją i możemy użyć konstrukcji PRINT VAL$(a$ + "$").

Niestety nie wszystkie funkcje działają w pełni zgodnie intencjami autorów interpretera ZX—Basic. W rozdziale "Błędy w systemie" podajemy znane niedoskonałości w działaniu funkcji SCREEN$, STR$ i USR.

Instrukcje

ZX-Basic stawia do naszej dyspozycji 50 instrukcji. Wszystkie one mogą być umieszczone w programie lub wykonane natychmiast jako rozkazy wydane z klawiatury, choć dla niektórych z nich tylko jedno z tych zastosowań może być sensowne i korzystne.

BEEP x,y

Wbudowany mały głośniczek pozwala generować dźwięki. Parametr x oznacza czas trwania dźwięku w sekundach i musi zawierać się w przedziale —0.5<=x<10.5. Drugi parametr y określa wysokość tonu. Wartość y=0 odpowiada środkowemu C. Zakresem zmienności y jest —60...69.8, co odpowiada mniej więcej trzem oktawom. Przekroczenie dopuszczalnych zakresów sygnalizowane jest komunikatem B. W czasie generowania tonu nie jest kontrolowany klawisz BREAK, w związku z czym przerwanie programu jest możliwe dopiero po zakończeniu wykonywania się tej instrukcji.

BORDER k

Komenda ta określa kolor ramki na ekranie telewizyjnym. Parametr k powinien mieścić się w przedziale 0<= k <=7 i oznacza numer wybranego koloru. Przekroczenie przez k dopuszczalnego zakresu jest sygnalizowane komunikatem B. Rozkazem BORDER k ustawia się również kolor tła w dolnej części ekranu przeznaczonej na komunikaty systemowe i do wprowadzania danych instrukcją INPUT.

BRIGHT k

Każdy kolor tła może występować w dwóch odcieniach: normalnym i rozjaśnionym. k=0 określa odcień tła jako normalny, k=1 jako rozjaśniony, k=8 oznacza zaś stosowanie w danym polu odcienia już w nim określonego. BRIGHT k może występować jako samodzielna instrukcja lub w ramach PRINT lub INPUT, obowiązując wówczas jedynie w czasie wykonywania danej instrukcji. Równoważny efekt wywołuje wydruk CHR$ 19 + CHR$ k. Symbole te można wprowadzać zarówno do tekstu programu, jak i do stałych tekstowych, naciskając w trybie rozszerzonym (kursor E) klawisz 8 jako BRIGHT 0 i klawisz 9 jako BRIGHT 1.

CAT...

Instrukcja ta może być stosowana jedynie z podłączonym ZX Interface 1.

CIRCLE [c,] k,m,n

Rozkaz rysowania okręgu o środku w punkcie o współrzędnych (k,m) i promieniu n w kolorach określonych przez ciąg c. Pominięcie c w liście parametrów powoduje przyjęcie domyślnych wartości INK 8, PAPER 8, BRIGHT 8 oraz FLASH 8. Dopuszczalne zakresy dla parametrów to: 0<= k<=255, 0<= m <=175. n trzeba tak dobierać, by nie dochodziło do prób rysowania poza ekranem, w przeciwnym wypadku wystąpi błąd B.

CLEAR [k]

Rozkaz porządkujący system. Czyści obszar zmiennych, zwalniając zajmowaną przez nie pamięć, czyści ekran oraz stos adresów powrotnych z podprogramów. Ponadto wykonuje instrukcję RESTORE oraz zeruje wskaźniki określające bieżące współrzędne na ekranie zarówno dla instrukcji piszących, jak i rysujących. Jeżeli ponadto podany jest parametr k, jego wartość będzie przyjęta jako nowy RAMTOP. k musi mieścić się w przedziale 23821...65535, inaczej pojawi się komunikat M (Ramtop no good). Gdy w pamięci komputera już jest program wartość minimalna k musi być odpowiednio wyższa. Zbyt niskie umieszczenie RAMTOP może uniemożliwić wykonanie jakiejkolwiek instrukcji z klawiatury bądź z programu.

CLOSE # k

Instrukcja ta odłącza k—ty strumień od przypisanego mu kanału. Co prawda k może przyjmować wartości od 0 do 15, ale w praktyce próby odłączania strumieni od 0 do 3 są przez system ignorowane. Więcej informacji o kanałach i strumieniach można znaleźć w rozdziale "Kanały i strumienie", warto też przeczytać rozdział "Błędy w systemie".

CLS

Bezparametrowy rozkaz czyszczący ekran i zerujący wskaźniki ekranowe instrukcji piszących i rysujących. Po oczyszczeniu ekran przybiera kolor określony przez ostatnią instrukcję PAPER oraz BRIGHT.

CONTINUE

Instrukcja ta pozwala na wznowienie programu po jego zatrzymaniu. Jeżeli przerwanie nastąpiło w wyniku błędu sygnalizowanego komunikatem różnym od 9 lub L, to po CONTINUE zostanie powtórzona ostatnio wykonywana instrukcja, w pozostałych przypadkach wykonane będą następne instrukcje. Pozwala to na poprawianie z klawiatury sygnalizowanych błędów i kontynuowanie pracy. Instrukcja ta nie stosuje się do poleceń wydawanych z klawiatury. Nie ma sposobu na nakłonienie Spectrum do kontynuowania przerwanej sekwencji rozkazów nie umieszczonych w programie. Lokowanie CONTINUE w programie nie jest celowe, gdyż jej działanie będzie uzależnione od chwilowej zawartości zmiennych systemowych OLDPPC oraz OSPPC. Te z kolei są modyfikowane przez system i panowanie nad ich zawartością może być kłopotliwe, szczególnie podczas testowania programu.

COPY

Jeśli do Spectrum jest dołączona drukarka, po tym rozkazie zawartość górnych 22 linii ekranu zostanie skopiowana na papier (kopiowane są nie tylko napisy, ale również wszelkie rysunki). Przy braku drukarki instrukcja ta jest po prostu ignorowana. Spectrum potrafi samo rozpoznać, czy dołączono do niego drukarkę. W czasie kopiowania ekranu zatrzymywany jest wewnętrzny zegar. Zmienne systemowe FRAMES nie są w tym czasie modyfikowane.

DATA e1,e2,....

Niedoceniana instrukcja pozwalająca umieszczać w tekście programu listy danych. O jej użyteczności decyduje możliwość umieszczania na liście dowolnych wyrażeń, a nie tylko, jak w wielu innych dialektach Basic, stałych numerycznych i tekstowych. Prawidłowe jest więc użycie DATA w postaci:

100 DATA sin x + cos y +1, "wartość" + ("dodatnia" AND z>0) + ("ujemna" AND z<0)

Po rozkazie READ z, z$ zmienna z przyjmie wartość sin x+cos y+1, obliczoną dla bieżących wartości zmiennych x i y, zmienna zaś z$ będzie zawierała łańcuch "wartość dodatnia" lub "wartość ujemna", zależnie od znaku dopiero co wyznaczonej wartości z. Trzeba tylko pamiętać, by wszystkie zmienne występujące w wyrażeniach e1,e2,... miały w momencie wykonywania odpowiednich komend READ określone wartości.

Niecelowe jest wydawanie rozkazu DATA bezpośrednio z klawiatury, gdyż Spectrum po sprawdzeniu poprawności językowej natychmiast o nim zapomni. W programie może znajdować się dowolna liczba instrukcji DATA i można je umieścić w dowolnych miejscach.

DEF FN @[$](@$_1$[$],@$_2$[$], ...,@$_i$[$]) = e

Definiowanie funkcji przez użytkownika. Nazwą funkcji może być tylko jedna litera, ewentualnie ze znakiem $. Umieszczenie go określa funkcję o wartościach tekstowych, a pominięcie o wartościach numerycznych. Po nazwie, w obowiązkowych nawiasach, można podać listę argumentów oddzielanych przecinkami. Dopuszczalne są definicje funkcji bezargumentowych. Nazwy argumentów muszą być jednoliterowe (ewentualnie ze znakiem $). Ogranicza to maksymalną liczbę argumentów do 52 (26 numerycznych i 26 alfanumerycznych).

Po zamknięciu listy parametrów nawiasem i znakiem równości następuje właściwa definicja funkcji. Może nią być dowolne wyrażenie odpowiedniego typu, złożone ze stałych, zmiennych, argumentów, funkcji standardowych i innych funkcji zdefiniowanych przez programistę. Należy wystrzegać się pośredniego lub bezpośredniego wywołania funkcji przez samą siebie. Spectrum nie rozpozna tego jako błędu i jedynie po efektach będzie można się zorientować, że coś jest nie tak: po zapełnieniu pamięci wystąpi komunikat błędu (4 lub 6) lub nawet dojdzie do załamania się systemu.

Instrukcji DEF FN nie ma sensu wydawać z klawiatury, bo po sprawdzeniu poprawności byłaby natychmiast zapomniana. Można natomiast z klawiatury wywoływać funkcje umieszczone w tekście programu, nawet jeśli program nie był ani razu uruchamiany. Rozmieszczenie definicji w programie i ich kolejność nie ma znaczenia. W przypadku kilkakrotnego definiowania w jednym programie funkcji o tej samej nazwie i tego samego typu, definicją obowiązującą będzie wersja umieszczona w linii o najniższym numerze.

Zwróćmy uwagę, że w definicji funkcji mogą wystepować nazwy zmiennych używane w innych miejscach programu. Jedynie zmienne o nazwach identycznych z nazwami parametrów stają się dla obliczanej funkcji niewidoczne i ich role przejmują wartości poszczególnych parametrów. Wyznaczanie wartości funkcji nie ma wpływu na wartość jakiejkolwiek zmiennej, ani żadnej nie inicjuje.

DIM @[$]($k_1$,...,$k_j$)

Deklaracja zmiennej tablicowej. Rozkaz ten rezerwuje miejsce w pamięci na tablicę numeryczną lub znakową. Liczba wymiarów tablicy ograniczona jest do 255. Poszczególne indeksy określają ograniczenia górne, ograniczeniem dolnym jest zawsze liczba 1.

Rozkaz ten usuwa z pamięci istniejącą tablicę o tej samej nazwie i tego samego typu. Może więc być stosowany w programie wielokrotnie. Wartości definiujące wymiary muszą być znane w momencie wykonywania DIM, ale niekoniecznie już w czasie uruchamiania programu. Możliwe jest zatem deklarowanie tablic, wymiary których dopiero określi użytkownik. W trakcie pisania programu musi być określona jedynie liczba wymiarów. Nazwa tablicy znakowej nie może pokrywać się z nazwą prostej zmiennej tekstowej. Tablice numeryczne w momencie deklarowania są automatycznie wypełniane zerami, znakowe zaś symbolem spacji (kod 32).

Elementami tablicy znakowej są pojedyncze symbole. Tablice takie mogą jednak być traktowane jako tablice prostych zmiennych tekstowych o jednakowej i z góry zadanej długości. Długość ta jest ustalona przez ostatni wymiar definiujący tablicę. Odwoływanie się do tablicy znakowej jako do zbioru prostych zmiennych tekstowych następuje przez pomijanie ostatniego indeksu. Jeśli w programie zadeklarowano DIM A$(10) : DIM B$(5,20), to A$ = A$() jest zmienną alfanumeryczną o długości 10 oraz B$(i) = B$(i,) są pięcioma zmiennymi o długości 20. Istotną różnicą w porównaniu z prawdziwymi prostymi zmiennymi znakowymi jest to, że w powyższym przykładzie po instrukcji LET A$ = "ALA" długość A$ w dalszym ciągu wynosi 10. Brakujące symbole są automatycznie uzupełniane spacjami.

DRAW [c,] k,m [,x]

Instrukcja ta pozwala na rysowanie odcinków prostych (x opuszczone lub równe 0) oraz fragmentów łuku koła o kącie środkowym x wyrażonym w radianach. Początkiem rysowania jest punkt (x_0,y_0), w którym zakończyła rysowanie ostatnia instrukcja rysująca (PLOT, DRAW lub CIRCLE), a końcem punkt (x_0 + k,y_0 +m). Punkt (x_0,y_0) nie jest stawiany. Jeśli x>0, to łuk jest rysowany w kierunku przeciwnym do ruchu wskazówek zegara, a dla x<0 w kierunku zgodnym. Zastosowany algorytm działa zgodnie z oczekiwaniami jedynie dla małych wartości x. Przy większych wyniki mogą być efektowne, choć nieoczekiwane. Ciekawym polecamy sprawdzenie, co się stanie w wyniku wydania poniższych rozkazów:

PLOT 55,27 : DRAW OVER 1,120,120,59 ↑ 3*PI

ERASE...

Rozkaz działa jedynie z dołączonym ZX Interface 1.

FLASH k

Parametr k może, podobnie jak dla instrukcji BRIGHT, przyjmować jedynie wartości 0,1 lub 8. Rozkaz ten określa, czy pole na ekranie ma

migać (k=1), czy nie (k=0). k=8 oznacza, że status danego pola ma pozostać niezmieniony. FLASH jako samodzielna instrukcja określa sposób wyświetlania w obrębie całego programu, umieszczony zaś na liście PRINT lub INPUT jedynie podczas wykonywania tej konkretnej instrukcji.

FOR @ = x TO y [STEP z]

Otwarcie pętli: wykonuj dla @ równego od x do y z krokiem równym z, czyli zwiększaj (lub zmniejszaj dla z<0) każdorazowo wartość @ o z, aż będzie większa (lub mniejsza dla z<0) od y.

Rozkaz ten jest ściśle związany z opisaną dalej komendą NEXT @. Zadaniem instrukcji FOR jest utworzenie zmiennej sterującej (po uprzednim usunięciu z pamięci zmiennych numerycznych lub sterujących o tej samej nazwie). Zmienna taka przybiera w pamięci postać odmienną od zmiennych numerycznych: poza jej wartością przechowywane są wartości parametrów y, z oraz numer linii i instrukcji FOR kreującej @ (patrz rozdział "Wykorzystanie pamięci"). Po jej skonstruowaniu następuje kontrola, czy pętla może być przynajmniej raz wykonana. Jeśli tak, sterowanie jest przekazywane do następnej po FOR komendy. Jeśli nie, szuka się w programie komendy NEXT @, by oddać sterowanie do rozkazu występującego bezpośrednio po NEXT @. Tylko w tym wypadku Spectrum wykrywa ewentualny brak w programie instrukcji NEXT @ /komunikat I/. Pominięcie STEP spowoduje przyjęcie wartości domyślnej z=1.

Mimo, że ZX—BASIC formalnie nie odróżnia liczb zmiennoprzecinkowych od całkowitych, to dla x, y, z całkowitych z przedziału —65535...65535 obsługa pętli przyspiesza się o około 20%.

FORMAT...

Instrukcja działa jedynie po przyłączeniu ZX Interface 1.

GO SUB k

Wywołanie podprogramu. Rozkaz ten umieszcza na specjalnym stosie swój własny adres, a następnie przekazuje sterowanie do linii o numerze k lub pierwszym większym i wykonuje kolejne instrukcje aż do natrafienia na komendę RETURN, po której zdejmowany jest ze stosu adres ostatniego GO SUB i sterowanie oddane do rozkazu następującego bezpośrednio po nim. Pozwala to na wykonywanie tej samej sekwencji komend w różnych miejscach programu bez konieczności wielokrotnego ich przepisywania. Nie ma ograniczeń co do głębokości kolejnych wywołań podprogramów. Dopuszczalna jest również rekurencja, a więc wywoływanie procedury przez samą siebie.

GO TO k

Skok do linii o numerze k lub następnej. Jeśli nie ma linii k ani żadnej dalszej program kończy się komunikatem 0. Rozkaz ten wydany z klawiatury uruchamia program od zadanej linii. Uruchamianie programu przez GO TO ma tę przewagę nad RUN, że nie powoduje żadnych skutków ubocznych, w szczególności utraty wartości zmiennych, co może mieć katastrofalny wpływ na uruchamiany program.

IF x **THEN** s

Instrukcja warunkowa. Jeśli wartość wyrażenia x jest różna od zera wykonywany jest ciąg rozkazów s, w przeciwnym razie wszystkie komendy do końca linii są pomijane i sterowanie przechodzi do następnej linii. W praktyce x jest najczęściej wyrażeniem logicznym. Ponieważ jednak Spectrum takie wyrażenie traktuje jako numeryczne, możliwe jest stosowanie obu typów wymiennie. Jeśli w programie istnieje prosta zmienna numeryczna a, to poniższe dwie postacie rozkazu IF są równoważne

IF a<>0 THEN s
IF a THEN s.

INK k

Określenie koloru kolejnych znaków na ekranie. Rozkaz ten określa kolor atramentu. Parametr k może przyjmować jedną z wartości 0...9. Liczby od 0 do 7 oznaczają numer wybranego koloru, 8 sygnalizuje zgodę na stosowanie kolorów zastanych już w danym polu na ekranie, 9 nakazuje użycie koloru białego lub czarnego zależnie od tego, który z nich będzie bardziej kontrastował z ustalonym w danym polu kolorem tła. W dolnych liniach ekranu system zawsze stosuje INK 9. Instrukcja ta może występować samodzielnie w linii lub może być umieszczona na liście PRINT, INPUT, CIRCLE itd.

INPUT...

Pozwala podczas wykonywania programu wprowadzać z klawiatury dane przyporządkowywane wymienionym zmiennym. Rozkaz ten umożliwia również pisanie w dolnej części ekranu, napisy te utrzymują się jednak jedynie w czasie jego wykonywania i po zakończeniu są natychmiast kasowane.

Po słowie kluczowym INPUT umieszcza się listę elementów precyzujących co i jak ma być wykonywane. Elementy listy muszą być oddzielane separatorami: przecinkami, średnikami lub apostrofami. Przecinek nakazuje przesunąć kursor do następnej połowy ekranu w tej samej linii, a jeśli to niemożliwe, to do początku następnej linii. Apostrof oznacza przejście do nowej linii, a średnik nie zmienia położenia kursora. Elementy listy przewidziane do wyświetlenia muszą być stałymi lub wyrażeniami zamkniętymi w nawiasy okrągłe. Specyfikatory koloru INK, PAPER, BRIGHT, FLASH, INVERSE i OVER pozwalają na uzyskanie dowolnych efektów kolorystycznych. Zmienne, których wartości mają być podane z klawiatury powinny być umieszczane na liście pojedynczo między separatorami. Jeśli chcemy by program wczytał z klawiatury element tablicy A (i,j) i by poinformował użytkownika, czego od niego oczekuje, to skorzystamy z instrukcji:

INPUT "Podaj wartość elementu A(";(i);",";(j);")=";A(i,j)

Instrukcja INPUT czeka na wprowadzenie informacji i kończy się dopiero po podaniu wartości dla wszystkich zmiennych z listy. Każdą wartość z osobna trzeba kończyć wciśnięciem klawisza ENTER. Przy wczytywaniu wartości zmiennych tekstowych wciśnięcie samego ENTER spowoduje nadanie odpowiednim zmiennym wartości łańcucha pustego. Pytania o zmienną numeryczną nie można w ten sposób zignorować i trzeba podać jakąś liczbę, posługując się dowolnymi wyrażeniami, istniejącymi zmiennymi i stałymi. Przy wczytywaniu wartości zmiennych numerycznych na dole ekranu miga kursor L lub C, natomiast przy wprowadzaniu ciągu znaków kursor pojawi się wraz z otaczającymi go cudzysłowami. W razie potrzeby można je wykasować. Istnieje możliwość wczytywania łańcucha znaków bez tych cudzysłowów. Odpowiednią zmienną tekstową należy wtedy poprzedzić na liście słowem kluczowym LINE. Pojedyncze LINE odnosić się będzie tylko do jednej zmiennej. Niedopuszczalne jest stosowanie tego słowa wraz ze zmiennymi numerycznymi.

W trakcie wykonywania instrukcji INPUT dwie dolne linie ekranu zaczną się w razie potrzeby przesuwać do góry robiąc miejsce na dalsze wydruki i wprowadzane dane. Obszar ten może się maksymalnie rozszerzyć do 22 linii.

Wprowadzanie danych odbywa się w zasadzie za pomocą edytora. Dzięki temu przed naciśnięciem ENTER możliwe są wszelkie poprawki wprowadzanych wielkości.

Na liście INPUT można ponadto umieszczać TAB n oraz AT k,n. Działanie pierwszego z nich powoduje przesunięcie kursora do n—tej kolumny w danej lub następnej linii. Z kolei AT umieści kursor w położeniu k,n, to znaczy w k—tym wierszu i n—tej kolumnie. Nieco kłopotu może sprawiać fakt, że położenie początku układu współrzędnych — pola (0,0) — zmienia się wraz z rozszerzaniem się dolnej części

ekranu. Wskazuje ono zawsze lewy górny róg obszaru roboczego edytora i względem niego jest ustalane położenie pola k,n. Próby zmuszenia Spectrum do rozszerzenia tego obszaru ponad 22 linie kończą się sygnalizowaniem błędu 5. Przedtem jeszcze komputer może wyrażać swoją dezaprobatę buczeniem.

INVERSE k

Kolejny specyfikator kolorów. Parametr k może przyjmować jedynie wartości 0 i 1. 0 oznacza wyświetlanie obowiązujących kolorów tła i atramentu, 1 zaś zamienia je rolami. Jak wszystkie specyfikatory kolorów, tak i ten może występować jako samodzielna instrukcja w linii bądź być umieszczony na liście odpowiedniej instrukcji piszącej lub rysującej.

LET v=e

Najczęściej używana instrukcja, pozwalająca nadawać wartości poszczególnym zmiennym. Wykonanie jej polega na obliczeniu wyrażenia e i podstawieniu otrzymanej wartości pod v, jeśli taka zmienna już istnieje. Jeśli v jeszcze nie ma, to jest tworzona w odpowiednim obszarze. Nie dotyczy to tablic, które muszą być deklarowane przed użyciem. Jeżeli zmienna v ma w czasie wykonywania rozkazu określoną wartość, to może być również wykorzystywana do obliczania wyrażenia e.

LIST [k]

Rozkaz przeznaczony do wyświetlania na ekranie tekstu programu w języku Basic. Parametr k określa, od której linii ma być być drukowany program. Pominięcie k powoduje przyjęcie wartości domyślnej 0. Ponadto komenda ta ustawia wskaźnik linii bieżącej na linii k czyniąc ją dostępną dla edycji. Wyświetlanie odbywa się zawsze do końca tekstu programu. Przerwać je można po pojawieniu się w dolnej części ekranu pytania scroll? przez naciśnięcie jednego z klawiszy — BREAK, STOP lub "n".

LLIST [k]

Instrukcja analogiczna do poprzedniej, program jest jednak drukowany na dołączonej drukarce. Gdy drukarki brak LLIST ustawia jedynie wskaźnik linii bieżącej na linii k. Podczas wydruku programu zatrzymany jest wewnętrzny zegar (zmienne systemowe FRAMES nie są modyfikowane).

LOAD...

Spectrum może wczytywać z kasety magnetofonowej różne typy zbiorów.

LOAD f powoduje wczytanie z taśmy programu w języku Basic nagranego na taśmie jako zbiór o nazwie f wraz ze zmiennymi. Wszystkie inne zbiory z kasety są pomijane. Zamiast pełnej nazwy poszukiwanego zbioru można podać łańcuch pusty: LOAD "". Zostanie wówczas wczytany pierwszy znaleziony na taśmie program w języku Basic niezależnie od jego nazwy. Po znalezieniu na taśmie poszukiwanego bloku istniejący w pamięci program i zmienne są kasowane logicznie (są modyfikowane odpowiednie zmienne systemowe i tylko niewiele komórek pamięci zmienia swoją zawartość) i jest przygotowywane miejsce na nowy program. Ewentualne pozostałości po starym są dla systemu niedostępne.

LOAD f DATA @[$] [( )] wczytuje tablicę numeryczną lub znakową i umieszcza ją w obszarze zmiennych pod nazwą @, usuwając uprzednio zmienną tablicową o tej samej nazwie i tego samego typu. @ nie musi być tą samą nazwą pod jaką występowała dana tablica przed nagraniem na kasetę. Wymiarów tablic nie podaje się, ale nawiasy są obowiązkowe. Jako tablice tekstowe można wczytywać i nagrywać proste zmienne tego typu.

LOAD f CODE [k[,n]]. Postać rozkazu pozwalająca wczytywać zbiory bajtów bez sprecyzowanej struktury. Parametr k mówi, pod jaki adres ma być wczytany blok, a n określa jego długość. Jeśli parametry te pominąć, to zbiór zostanie wczytany w to samo miejsce, z którego został zrzucony na taśmę. Instrukcja ta nie sprawdza, czy wczytywany blok nie niszczy informacji żywotnej dla systemu i bezwzględnie zamazuje odpowiednie obszary pamięci nową informacją. Podanie wartości parametru n mniejszej od faktycznej długości zbioru na taśmie sygnalizowane jest komunikatem R, a więc m.in. nie można tą instrukcją czytać jedynie początkowych fragmentów bloków (chyba że wyłączając magnetofon w trakcie czytania).

LOAD f SCREEN$ jest szczególnym przypadkiem LOAD f CODE i odpowiada dokładnie LOAD f CODE 16384,6912, wczytując ekran wraz z atrybutami.

Przy czytaniu danych i programów z kasety są możliwe dwa rodzaje błędów. Problemy techniczne sygnalizowane są komunikatem R. Przy wczytywaniu zbyt długich bloków (częsty przypadek, gdy RAMTOP jest za nisko ustawiony) pojawia się komunikat 4.

LPRINT...

Instrukcja działająca jak PRINT, lecz dane wysyłane są nie na ekran, lecz na drukarkę. Gdy drukarki brak rozkaz jest ignorowany. Umieszczone na liście LPRINT specyfikatory kolorów PAPER, INK, FLASH i BRIGHT będą ignorowane. Separatory oraz TAB działają normalnie, ale w AT pierwszy parametr jest pomijany. Wydruk odbywa się za pośrednictwem bufora o długości 32 znaków (256 bajtów). Faktyczny wydruk następuje po zapełnieniu bufora, poleceniu przejścia do nowej linii oraz w momencie prawidłowego zakończenia się programu. Po przerwaniu programu na skutek błędu lub interwencji użytkownika wydruk niepełnego bufora następuje po naciśnięciu ENTER.

MERGE f

Wczytanie z taśmy programu w języku Basic i dołączenie go do istniejącego w pamięci. Linie, których numery w obu programach się pokrywają zostają zastąpione przez nowe, podobnie jak zmienne o pokrywających się nazwach. Pozostałe linie i zmienne są dopisywane w odpowiednich obszarach. Pozwala to na tworzenie bibliotek podprogramów do późniejszego wykorzystywania w innych programach.

Działanie instrukcji polega na wczytaniu całego bloku do obszaru roboczego i dopiero po tym przeglądane są stary i nowy program oraz dokonuje się ich połączenie. Przy dłuższych programach proces ten może trwać nawet kilka minut udając w tym czasie załamanie systemu (nic się nie dzieje na ekranie i komputer nie reaguje na żadne klawisze).

Rozkaz może być używany również wtedy, gdy w pamięci nie ma żadnego programu. Różni się wtedy od LOAD f znacznie dłuższym czasem wykonania oraz tym (cenne dla piratów), że nie uruchamia wczytanego programu nawet jeśli był nagrany przez SAVE f LINE k. Podobnie jak dla LOAD jako nazw można podać łańcuch pusty.

MOVE...

Kolejna instrukcja rozpoznawana jedynie przez Spectrum z dołączonym ZX Interface 1.

NEW

Instrukcja działająca prawie jak wyłączenie zasilania. Inicjuje system Basic zachowując jedynie zmienne systemowe RAMTOP, P RAM,

RASP, PIP i UDG. Pamięć poniżej RAMTOP jest fizycznie czyszczona. Nietknięty pozostaje natomiast obszar powyżej RAMTOP, na ogół zachowane są więc symbole definiowane przez użytkownika (o ile on sam nie ulokował ich gdzieś poniżej RAMTOP).

NEXT @

Instrukcja zamykająca pętlę otwartą przez FOR... Po natrafieniu na NEXT @ interpreter odszukuje zmienną sterującą @, powiększa jej wartość o wielkość kroku z i porównuje z wartością końcową y. Jeśli ta nie została jeszcze przekroczona (w górę lub w dół zależnie od znaku z) to sterowanie przekazywane jest do instrukcji następnej po FOR, które jako ostatnie zainicjowało @. W przeciwnym razie wykonywane będą kolejne instrukcje za NEXT @. Nieznalezienie w obszarze zmiennych żadnej zmiennej numerycznej o nazwie @ sygnalizowane jest komunikatem 2. Jeśli jednak jest tam zmienna, ale nie będąca zmienną sterującą, to pojawi się komunikat 1.

Taka realizacja pętli ma różne konsekwencje. Po pierwsze jednocześnie może być otwartych maksimum 26 pętli. Zmienne sterujące mogą być wykorzystywane jako zwykłe zmienne numeryczne zarówno wewnątrz pętli jak i poza nią. Po otwarciu pętli nie można modyfikować ani wartości końcowej ani kroku (nie dlatego, że nie wolno, ale jest to po prostu z poziomu BASIC niemożliwe, bez skomplikowanego grzebania w obszarze zmiennych).

Po drugie, nie ma ograniczeń co do sposobu wzajemnego położenia pętli. Np. poniższa konstrukcja, niedopuszczalna w innych wersjach Basic, będzie potraktowana przez Spectrum jako prawidłowa i wykonana zgodnie z zasadami, choć może nie całkiem zgodnie z intencjami programisty:

```
100 FOR I = 1 TO 10
200 FOR J = 10 TO 20
300 PRINT I,J
400 NEXT I
500 NEXT J
```

Warto program ten wykonać na własnym komputerze i zastanowić się, dlaczego daje takie dziwaczne wyniki. W pewnych sytuacjach bywa również możliwy skok do wnętrza pętli, co jednak należy traktować jako osobliwość tej konkretnej realizacji języka i w praktyce unikać takich sztuczek.

OPEN #k

Specyfikator kanału Rozkaz pozwalający przyłączyć k—ty strumień do wskazanego kanału (patrz rozdział "Kanały i strumienie"). "Gołe" Spectrum, (bez urządzeń zewnętrznych, a konkretnie bez Interface 1) rozpoznaje jedynie kanały "S", "K", "P", "s", "k", "p". Nie należy ruszać strumieni 0...3, pozostałe 4...15 są do dyspozycji użytkownika.

OUT m,n

Jest to odpowiednik rozkazów asemblera:

```
LD A,n
LD BC,m
OUT (C),A
```

m powinno być liczbą z przedziału 0...65535, n zaś z —255...255, przy czym liczby ujemne będą automatycznie powiększane o 256 (tak więc OUT m,—3 to to samo, co OUT m,253).

Instrukcja ta pozwala z poziomu Basic wysyłać informacje do portów I/O procesora i w zasadzie może mieć zastosowanie przy współpracy ZX Spectrum z urządzeniami zewnętrznymi. W "gołym" komputerze jej możliwości ograniczają się do wzbudzania głośnika oraz chwilowego ustawiania koloru ramki ekranu.

OVER k

Specyfikator sposobu nakładania się punktów lub symboli. k=0 oznacza zastępowanie istniejącego punktu lub znaku przez nowy, a k=1 powoduje nakładanie się znaków nowych na już istniejące: w miejscach, w których stary i nowy znak lub punkt mają te same kolory pozostaje kolor tła, tam zaś, gdzie kolory są różne pojawia się kolor atramentu. Nowy rysunek powoduje więc "odwracanie" koloru dotychczasowego. Pozwala to na kasowanie pojedynczych kropek, linii na ekranie itp. Podobnie jak dla specyfikatorów kolorów zakres obowiązywania danego trybu rysowania może być globalny w całym programie lub tylko w czasie wykonywania jednej instrukcji piszącej czy rysującej.

PAPER k

Specyfikator określający kolor tła. Wartości i znaczenia parametru k są takie same, jak dla instrukcji INK. W odróżnieniu od INK kolor tła może występować w dwóch różnych odcieniach (patrz BRIGHT).

PAUSE k

Czasowe wstrzymanie działania programu. Parametr k określa ile pięćdziesiątych sekundy ma trwać przerwa. Naciśnięcie dowolnego klawisza przerywa pauzę niezależnie od upływu czasu. Wartość k=0 oznacza pauzę trwającą aż do naciśnięcia któregokolwiek klawisza (patrz również rozdział "Błędy w systemie").

PLOT [c,] k,m

Rysowanie pojedynczego punktu o współrzędnych 0<= ABS k <= 255 oraz 0<= ABS m <= 175 na ekranie. Początek układu współrzędnych jest umieszczony w lewym dolnym rogu, powyżej dwóch linii zarezerwowanych dla komunikatów systemowych. Rozkaz ten nie może być stosowany w dolnych dwóch wierszach.

POKE k,m

Instrukcja ta do komórki o adresie 0<= k <= 65535 ładuje liczbę —255<= m <= 255 (powiększoną o 256 dla m<0). Modyfikowanie komórek o adresach mniejszych od 16384 (obszar ROM) jest oczywiście nieskuteczne.

PRINT...

Podstawowa instrukcja do wyświetlania na ekranie ciągów znaków i liczb. PRINT elementy ze swej listy drukuje w górnych 22 liniach ekranu. Na liście można umieszczać to samo, co i na listach INPUT. Nie ma potrzeby zamykania w nawiasy wyrażeń, które przed wydrukowaniem mają być obliczone. Łatwiej zastosować AT k,n, gdyż położenie początku układu współrzędnych jest stałe. Mieści się on w lewym górnym rogu ekranu. Parametr k określa linię (0<= k <= 21) a m — kolumnę (0<= m <= 31). Przekroczenie przez k lub m dopuszczalnych zakresów sygnalizowane jest komunikatem 5.

Po zapełnieniu 22 wierszy ekranu przed dalszym drukowaniem komputer pyta użytkownika (scroll?), czy może przesunąć zawartość ekranu do góry, by zrobić miejsce na dalsze informacje. Klawisze BREAK, STOP lub "n" przerywają wykonywanie programu z komunikatem D, pozostałe oznaczają zgodę na przesuwanie ekranu.

RANDOMIZE [k]

W zasadzie instrukcja ta jest przeznaczona do zainicjowania generatora pseudolosowego (RND). Jeżeli parametr k jest podany i różni się od zera zostaje on umieszczony w zmiennej systemowej SEED. Co się tam z nim dalej dzieje opisaliśmy przy okazji omawiania funkcji RND. Pominięcie parametru lub użycie 0 spowoduje umieszczenie w zmiennej SEED dwóch młodszych bajtów zmiennych systemowych FRAMES. Ubocznym zastosowaniem instrukcji RANDOMIZE może być wyznaczanie wartości młodszego i starszego bajtu danej liczby całkowitej k mieszczącej się w przedziale 1...65535. Młodszy bajt wyznaczamy przez PEEK 23670, a starszy przez PEEK 23671.

READ v1,v2,...

Rozkaz ten nadaje kolejnym zmiennym z listy wartości odpowiednich wyrażeń umieszczonych na listach DATA. Każda z instrukcji READ modyfikuje odpowiednią zmienną systemową (DATADD) pamiętającą, które wyrażenie z list DATA było ostatnio czytane. Próby dalszego czytania danych z tego źródła, po ich wyczerpaniu, są sygnalizowane komunikatem E.

REM

Jest to jedna z najcenniejszych, a niedocenianych instrukcji. Za słowem REM można umieścić dowolny ciąg znaków (z wyjątkiem symbolu o kodzie 13), który będzie przez komputer ignorowany. Rozkaz ten służy więc głównie do umieszczania w tekście programu komentarzy wyjaśniających różne związane z nim sprawy (np. do czego są używane poszczególne zmienne, co robi dany fragment itd.). Niedocenianie i niestosowanie komentarzy mści się proporcjonalnie do długości programu, który trzeba uruchomić lub zmodyfikować.

RESTORE [k]

Instrukcja ta ustawia wskaźnik systemowy na pierwsze wyrażenie na liście DATA w linii o numerze k lub pierwszym wyższym. Pominięcie k powoduje przyjęcie wartości domyślnej 0. Przed ponownym uruchomieniem programu instrukcją GO TO konieczne jest użycie rozkazu RESTORE (chyba, że w programie tym nie korzystamy z instrukcji READ).

RETURN

Rozkaz ściśle związany z GO SUB. W programie można umieścić dowolną ilość RETURN, ale nie można jej wykonać więcej razy, niż wykonano przedtem GO SUB. Spowodowałoby to przerwanie programu komunikatem 7.

RUN [k]

Instrukcja służąca do uruchamiania programów w języku Basic znajdujących się w pamięci komputera. Wykonywanie programu rozpoczyna się od linii o numerze k lub, jeśli k pominięto, od początku programu. Przed samym uruchomieniem RUN wykonuje jeszcze automatycznie rozkaz CLEAR ze wszystkimi jego konsekwencjami. Z uwagi na niszczenie przez CLEAR wszystkich istniejących zmiennych warto czasem przed użyciem RUN chwilę się zastanowić i... może sięgnąć po GO TO?

SAVE..

Instrukcja odwrotna do LOAD. Przy jej pomocy nagrywamy na kasetę programy w języku Basic, tablice znakowe lub numeryczne oraz zbiory bajtów bez precyzowania ich struktury. Postać rozkazu jest taka sama, jak LOAD z tym, że nie można pomijać żadnych parametrów. Nazwa,pod jaką dany blok ma zostać umieszczony na taśmie, musi być niepustym ciągiem znaków o długości nie przekraczającej 10. Nazwę można budować z dowolnych symboli.

Przy nagrywaniu programów w języku Basic po nazwie można umieścić LINE k. Tak nagrany program po późniejszym wczytaniu będzie się automatycznie uruchamiał od linii o numerze k.

STOP

Rozkaz ten pozwala zatrzymać program. Wznowić go można od następnej instrukcji za pomocą CONTINUE. STOP jest instrukcją przydatną zwłaszcza podczas testowania nowego programu.

VERIFY...

Postać tej instrukcji jest identyczna, jak LOAD... Niczego jednak nie wczytuje do komputera, a jedynie sprawdza, czy zawartość odpowiednich obszarów pamięci jest identyczna z tym, co nagrano na kasetę. W odróżnieniu od instrukcji LOAD, VERIFY nie zmienia zawartości pamięci. Wszelkie wykryte niezgodności są sygnalizowane komunikatem R.

Symbole kontrolne

Wśród symboli używanych przez Spectrum są tzw. symbole kontrolne. Mają one kody mniejsze od 32 (nie wszystkie z nich są wykorzystywane). Zastosowania ich na poziomie ZX—Basic są raczej ograniczone, gdyż te same efekty można uzyskiwać innymi instrukcjami. Są one jednak niezastąpione dla programujących w kodzie maszynowym. Poniżej przedstawiamy te z nich, których wydruk wywołuje jakiś efekt.

CHR$ 6 — Ten symbol może być stosowany wymiennie z przecinkiem jako separator na listach INPUT i PRINT.

CHR$ 8 — Jedyny cenny, z punktu widzenia programujących w języku Basic, symbol kontrolny. Powoduje cofnięcie kursora na ekranie o jedno pole w lewo (w czasie drukowania ciągów znaków).

CHR$ 9 — Patrz rozdział "Błędy w systemie".

CHR$ 13 — Jest to symbol klawisza ENTER. Na listach INPUT i PRINT może być stosowany zamiast pojedynczego apostrofu. W programach w ZX—Basic umieszczany jest on na końcu każdej linii.

CHR$ 16 — INK
CHR$ 17 — PAPER
CHR$ 18 — FLASH
CHR$ 19 — BRIGHT
CHR$ 20 — INVERSE
CHR$ 21 — OVER
CHR$ 22 — AT
CHR$ 23 — TAB

Symbole od CHR$ 16 do CHR$ 21 muszą być drukowane w postaci CHR$ k + CHR$ m lub CHR$ k ; CHR$ m, gdzie m jest wartością odpowiedniego dla danej instrukcji parametru. Inne separatory są niedopuszczalne.

Po symbolach CHR$ 22 i CHR$ 23 muszą wystąpić dwa inne w charakterze parametrów. Dla AT jest to zrozumiałe, bo trzeba podać numer wiersza i kolumny. Argumentem TAB jest jedna liczba, ale przedstawiana zawsze na dwóch bajtach. W praktyce wartość drugiego (starszego bajtu) nie ma znaczenia, bo i tak liczba ta jest brana modulo 32 (uwzględniana się tylko resztę z dzielenia przez 32). Pozostałe z wykorzystywanych symboli kontrolnych używane są przez edytor w trakcie czytania klawiatury i nie mają praktycznego znaczenia dla programisty. W czasie drukowania ciągów znaków symbole nie wymienione powyżej o kodach mniejszych od 32 są zastępowane znakiem zapytania.

Komunikaty systemowe

Każde wykrycie przez Spectrum błędu w programie powoduje jego przerwanie i wydruk odpowiedniego komunikatu w dole ekranu. Informacje te są zawsze w takiej samej standardowej postaci:

nr Tekst komunikatu k:m

Numer jest cyfrą 0...9, lub literą A...R. Tekst wyjaśnia po angielsku przyczynę przekazania sterowania do edytora. Liczby k:m informują, w której linii, która instrukcja wywołała odpowiedni komunikat. k=0 odnosi się zazwyczaj do rozkazów wydawanych bezpośrednio z klawiatury.

0 OK

Właściwe zakończenie komend wydawanych z klawiatury, wyczerpanie linii wykonywanego programu lub skok za pośrednictwem GO TO za linię o największym numerze. Przy obsłudze takiego "błędu" nie są

modyfikowane zmienne systemowe OLDPPC i OSPPC, w rezultacie czego rozkaz CONTINUE wydany po takim komunikacie spowoduje ponowne wykonanie ostatniej instrukcji w programie (nie dotyczy to rozkazów wydawanych z klawiatury).

1 NEXT without FOR

Interpreter natrafił na instrukcję NEXT @. W obszarze zmiennych nie znajduje się zmienna sterująca o danej nazwie, ale jest prosta zmienna numeryczna @. Gdyby i takiej nie było, to w analogicznej sytuacji zostałby wydrukowany komunikat 2. Najczęstszą przyczyną błędu jest pominięcie odpowiedniej instrukcji FOR lub skok do wnętrza pętli.

2 Variable not found

Próba użycia zmiennej nie istniejącej w obszarze zmiennych, a więc takiej, której nie nadano jeszcze żadnej wartości za pośrednictwem LET, READ, INPUT, FOR lub DIM.

3 Subscript wrong

Indeksy tablic przekroczyły dopuszczalne zakresy (ale zmieściły się w 0...65535), lub podano ich niewłaściwą liczbę, również próba wyznaczenia symbolu prostej zmiennej alfanumerycznej o numerze większym od długości zmiennej. Jeśli indeksy przekroczą wartość 65535, to w analogicznej sytuacji pojawi się komunikat B.

4 Out of memory

Brak miejsca w pamięci na wykonanie pożądanej akcji. Najczęściej następuje w czasie wykonywania instrukcji LET, INPUT, READ, DIM, GO SUB, LOAD, MERGE, próbie obliczenia funkcji zdefiniowanej rekurencyjnie. Na ogół przyczyną jest zbyt niskie położenie RAMTOP. Do wyjścia z kłopotu może okazać się konieczne skasowanie jakiejś linii w programie, tak, by zrobić miejsce umożliwiające podjęcie działań z klawiatury.

5 Out of screen

INPUT usiłuje zająć więcej niż 22 linie, lub parametry AT wskazują pole poza 22 górnymi wierszami ekranu.

6 Number too big

W czasie prowadzonych obliczeń usiłowano przekroczyć 1.7E38. Tym komunikatem kończą się próby dzielenia przez 0 lub obliczenia np. TAN (PI/2).

7 RETURN without GO SUB

Usiłowanie wykonania RETURN bez uprzedniego GO SUB, m.in. gdy liczba wykonanych GO SUB jest mniejsza od liczby wykonanych RETURN.

8 End of file

Ten komunikat może się pojawić tylko w ZX Spectrum z dołączonym ZX Interface 1.

8 STOP statement

Wykonano instrukcję STOP.

A Invalid argument

Podano niewłaściwy argument funkcji standardowej, np. próbując obliczyć SQR lub LN dla liczby ujemnej lub argument alfanumeryczny funkcji USR nie jest jedną właściwą literą. Ten błąd nie dotyczy argumentów funkcji definiowanych przez programistę.

B Integer out of range

Całkowitoliczbowy parametr instrukcji przekroczył dopuszczalny dla danego rozkazu zakres.

C Nonsens in Basic

Analizowany tekst jest niepoprawny z punktu widzenia reguł języka. Najczęściej pojawia się, gdy argumenty funkcji VAL lub VAL$ nie przedstawiają poprawnej postaci wyrażenia. Komunikat ten pojawi się również przy próbie wykonywania rozkazów nierozpoznawanych przez system, np. wczytano i uruchomiono program współpracujący z Microdrive'm, a nie podłączono ZX Interface 1.

D BREAK — CONT repeats

Przerwano działanie programu w czasie wykonywania instrukcji związanych z urządzeniem zewnętrznym (drukarką, magnetofonem lub telewizorem — po pytaniu scroll? naciśnięto BREAK, STOP lub "n"). Wydany z klawiatury rozkaz CONTINUE powtórzy jeszcze raz przerwany rozkaz.

E Out of DATA

Próba czytania danych z list DATA po ich wyczerpaniu.

F Invalid file name

Usiłowano nagrać na kasecie zbiór o nazwie pustej lub dłuższej niż 10 znaków.

G No room for line

Brak miejsca w pamięci na kolejną linię programu, gdyż jest on zbyt długi lub zbyt nisko ustawiono RAMTOP.

H STOP in INPUT

Program został przerwany w czasie wykonywania instrukcji INPUT.

I FOR without NEXT

Inicjowana pętla nie może być ani razu wykonana (np. FOR I=1 TO 0), a w programie nie ma instrukcji NEXT.

J INVALID I/O device

Błąd możliwy do popełnienia po przyłączeniu ZX Interface 1.

K Invalid colour

Argument instrukcji INK, PAPER, BORDER, BRIGHT, FLASH, OVER lub INVERSE przekroczył dopuszczalny dla danego rozkazu zakres. Komunikat ten może się również pojawić po wydrukowaniu symbolu kontrolnego sterującego kolorem, jeśli następny drukowany symbol ma kod będący niewłaściwym argumentem dla danego symbolu kontrolnego.

L BREAK into program

Przerwanie wykonywania programu przez użytkownika.

M RAMTOP no good

Próba nadania zmiennej systemowej RAMTOP zbyt małej lub zbyt dużej wartości.

N Statement lost

Próba wykonania za pomocą RETURN, NEXT lub CONTINUE skoku do nieistniejącej już instrukcji (np. wykasowanej w czasie poprawiania).

O Invalid stream

Próba przesyłania informacji za pomocą strumienia przed dołączeniem go do jakiegoś kanału lub o numerze większym od 15.

P FN without DEF

Wywołano niezdefiniowaną funkcję.

Q Parametr error

Niezgodność ilości lub typów argumentów przy wywoływaniu funkcji zdefiniowanej przez użytkownika.

R Tape loading error

Odpowiedni zbiór na kasecie został odnaleziony, ale z jakichś względów nie mógł być wczytany do komputera (na ogół przyczyną są powody techniczne).

Rozdział IV

ARYTMETYKA KOMPUTEROWA

Pojedyncza komórka pamięci komputerów opartych na procesorze Z80 mieści w sobie jedno słowo (bajt) 8—bitowe reprezentowane przez ciąg ośmiu cyfr, z których każda może być zerem lub jedynką. Interpretujemy je jako liczby w zapisie dwójkowym. Przejście od postaci binarnej do dziesiętnej jest bardzo proste. Wypiszmy w jednym rzędzie kolejno liczby 2 ↑ 7,2 ↑ 6,...,2 ↑ 1,2 ↑ 0=1. Następnie pod każdą z tych liczb wypiszmy jedną cyfrę przedstawienia dwójkowego. Dodając do siebie tylko te liczby z górnego rzędu, pod którymi znalazły się jedynki, otrzymujemy wartość dziesiętną. Zilustrujmy to przykładem przeliczenia liczby binarnej 01001101 na postać dziesiętną.

128	64	32	16	8	4	2	1
0	1	0	0	1	1	0	1
	64			8	4		1=77

Dzięki temu możemy myśleć o zawartości jednej komórki jako o liczbie dodatniej z przedziału 0...255.

Czasem jednak wygodniej jest rozpatrywać te liczby jako wartości ze znakiem. Możemy na przykład umówić się, że najstarszy bit (ten po lewej) określa znak (+, jeśli 0 i —, jeśli 1). Przy takiej interpretacji dysponujemy liczbami od —128 do 127. Problem jednak w tym, że komputer nie wie,jak w danej chwili chcemy traktować liczby, a działania potrafi wykonywać jedynie na liczbach naturalnych bez znaku. Żeby było jeszcze trudniej działania na liczbach 8-bitowych prowadzi się w ten sposób, że za wynik operacji uznaje się jedynie resztę z dzielenia przez 256 i co najwyżej zapamiętuje się fakt zaistnienia nadmiaru.

Rozwiązaniem przyjętym w Spectrum jest zastosowanie tzw. kodu uzupełnień do dwóch. Liczby mniejsze od zera zapisujemy w nim jako 256 + k. Np. liczbę —7 przedstawiamy jako 249 = 256—7. W tym systemie najstarszy bit również jest interpretowany jako znak liczby. Nie ma za to problemów z działaniami. Podstawowe działania arytmetyczne, jakim jest dodawanie, prowadzi do poprawnych wyników niezależnie od wyboru sposobu zapisu liczb. Np. 246+1 = 247 jest oczywistą równością dla liczb bez znaku, ale jest również poprawną równością jeśli myśleć o 246 jako o —7, bo wtedy 247 = 256+(—6) oznacza —6.

Posługiwanie się tak małymi wielkościami w wielu przypadkach jest niewystarczające. Większe liczby trzeba przechowywać w kilku kolejnych bajtach. W Spectrum bardzo ważną rolę pełnią liczby 16—bitowe. Cyfry ich dzieli się na dwie grupy po 8 bitów. Bardziej znaczącą grupę (tę od lewej) nazywamy starszym bajtem, a pozostałe młodszym. We wszystkich komputerach opartych na Z80 przyjęta jest zasada, że jeśli dwa kolejne bajty o adresach x, x+1 zawierają jakąś liczbę całkowitą to pod adresem x jest umieszczony młodszy bajt, a w następnej komórce starszy. Aby odtworzyć, jaką liczbę zawierają te komórki, trzeba starszy bajt pomnożyć przez 256 i dodać młodszy. Jeśli ma to być liczba ujemna przedstawiona w kodzie uzupełnień do dwóch, to jeszcze odejmujemy 2∧16 czyli 65536. Pozwala to operować na liczbach od 0 do 65535 lub od —32728 do 32727.

Przy programowaniu w kodzie maszynowym często zachodzi potrzeba wykonywania różnych operacji na pojedynczych bitach. Używanie zapisu dziesiętnego jest wtedy bardzo niewygodne ze względu na stałą konieczność przeliczania postaci binarnej na dziesiętną i odwrotnie. Z drugiej strony posługiwanie się 16—bitowymi liczbami szybko staje się koszmarem. Praca z nimi jest nie tylko pracochłonna, ale ze względu na swą monotonię szybko okazuje się źródłem licznych błędów.

Wybrano wyjście pośrednie, liczby o podstawie szesnastkowej (o ile 16 można uznać za wartość pośrednią między 2 i 10). Często nazywa się je heksadecymalnymi. Do ich zapisu używa się cyfr z zakresu 0,...,15. Wobec braku odpowiedników wśród liczb arabskich, dla liczb 10,...,15 stosuje się litery A,...,F. Przeliczanie postaci szesnastkowej na dziesiętną nie jest trudne, ale podręczny kalkulator będzie tu pomocny. Zasadą jest mnożenie kolejnych cyfr przez kolejne potęgi liczby 16 i dodawanie ich do siebie. Np. liczba heksadecymalna FFFF przedstawia 15*16 ↑ 3 + 15*16 ↑ 2 + 15*16 + 15 = 15*4096 + 15*256 + 15*16 + 15 = 65535.

Przejście od postaci dwójkowej do szesnastkowej jest szczególnie proste. Dzielimy cyfry liczby binarnej na grupy po cztery bity i każda taka czwórka przedstawia jedną cyfrę heksadecymalną. Np.

1001111110 = 0010 0111 1110 = 27E.

Jak widać **główną zaletą liczb heksadecymalnych jest duża łatwość w przechodzeniu do postaci binarnej i na odwrót,** przy jednoczesnej dużej zwartości zapisu. Wśród profesjonalnych programistów liczby te upowszechniły się na tyle, że wielu z nich traktuje umiejętność swobodnego posługiwania się nimi jako miarę stopnia opanowania języków asemblerowych.

Aby ułatwić Czytelnikom łagodne oswajanie się z liczbami szesnastkowymi w dalszej części tekstu będziemy wszędzie tam, gdzie liczby te będą miały znaczenie dla programujących w kodzie maszynowym, podawać je w dwu równoważnych postaciach: dziesiętnej i heksadecymalnej. **Liczby szesnastkowe poprzedzimy dla odróżnienia tradycyjnie przyjętym znakiem #.**

Sposoby kodowania liczb większych i zmiennoprzecinkowych są w różnych komputerach rozmaite. W ZX—Basic wszystkie liczby są przechowywane w pięciu kolejnych komórkach pamięci. Dla programisty istnieją tylko liczby zmiennoprzecinkowe, ale w pamięci liczby całkowite z przedziału —65536...65535 są kodowane w innej postaci niż pozostałe. Pierwszy i ostatni bajt w ich przedstawieniu zawsze zawiera zero. Drugi bajt jest równy 0 dla liczb dodatnich i 255 dla ujemnych. W trzecim i czwartym umieszcza się odpowiednio młodszy i starszy bajt danej liczby, przy czym wartości ujemne są pamiętane w kodzie uzu-

pełnień do dwóch. Np. liczba 38 wygląda tak: 0 0 0 38 0, natomiast 743 = 2*256 + 231 ma postać 0 0 231 2 0. Z kolei – 1231 zapamiętane jest jako 0 255 251 49 0, bo 256*251 + 49 = 65536 — 1231.

Z pozostałymi liczbami sprawa jest bardziej zawiła. Wykorzystuje się fakt, że każdą liczbę można jednoznacznie przedstawić w postaci 2 ↑ n*m, gdzie m jest liczbą z przedziału [1/2,1]. W pierwszym bajcie Spectrum przechowuje wartość n + 128. W pozostałych czterech umieszczona jest liczba m w postaci binarnej, a dokładnie — m*2 ↑ 32, z tym, że najstarszy bit, który powinien zawsze być równy 1 służy do przechowywania znaku liczby. Jedynka oznacza minus a zero plus.

Przypuśćmy, że pięć kolejnych komórek pamięci zawiera liczby 130 212 16 34 178. Są one interpretowane jako liczba —2 ↑ (130 — 128)* (212/2 ↑ 8 + 16/2 ↑ 16 + 34/2 ↑ 24 + 178/2 ↑ 32), natomiast bajty 106 112 4 231 78 przedstawiają liczbę 2 ↑ (106 — 128)*((128 + 112)/2 ↑ 8 + 4/2 ↑ 16 + 231/2 ↑ 24 + 78/2 ↑ 32).

Zaletą języków wysokiego poziomu, w tym ZX—Basic, jest uwolnienie programistów od konieczności zajmowania się sposobami przedstawiania różnych liczb w pamięci i przeliczeniami takimi jak wyżej. Jednak programując w asemblerze bez tego typu informacji nie moglibyśmy się obejść.

Rozdział V

WYKORZYSTANIE PAMIĘCI

Po włączeniu komputera do sieci automatycznie uruchamia się program umieszczony w ROM pod adresem 0. Jego zadaniem jest sprawdzenie ilości dostępnej pamięci, podział jej na różne bloki oraz nadawanie zmiennym systemowym ich początkowych wartości.

Poniższy diagram przedstawia podział pamięci. Obszary od 0 do 16383 mieszczą się w ROM. Pozostałe są w RAM. Tylko część bloków ma swoje ustalone położenie, inne mogą się przemieszczać w pamięci i ich aktualne adresy są przechowywane przez odpowiednie zmienne systemowe. Liczby na końcach niektórych obszarów oznaczają znaczniki końca i są tam zawsze obecne.

adres	obszar	znacznik końca
0 = #0000	Procedury systemowe	
15616 = #3D00	Wzorce symboli o kodach od 32 do 127	
16384 = #4000	Ekran	
22528 = #5800	Atrybuty	
23296 = #5B00	Bufor drukarki	
23552 = #5C00	Zmienne systemowe	
23734 = #5CB6	Mapa Microdrive'a	
CHANS	Informacje o kanałach	128 = #80
PROG	Program Basic	
VARS	Obszar zmiennych ZX-Basic	128 = #80
E_LINE	Bufor edytora	128 = #80
WORKSP	Bufor instrukcji INPUT Obszar roboczy ZX-Basic	13 = #0D
STKBOT STKEND	Stos kalkulatora	
sp	Wolny obszar pamięci dla ZX—Basic	
ERR_SP	Stos maszynowy	
RAMTOP	Stos adresów powrotnych GO SUB	62 = #3E
	Wolny obszar pamięci	
UDG P_RAMT	Wzory symboli definiowanych przez użytkownika	

Procedury systemowe (0 do 15615)

Temu obszarowi poświęcamy rozdział "Procedury systemowe".

Wzorce symboli (15616 do 16383)

Kształty symboli drukowanych o kodach od 32 do 127 (po 8 bajtów na każdy znak) zapełniają 768 ostatnich komórek ROM. Zakodowane są w analogicznej postaci jak symbole definiowane przez użytkownika. Znaki grafiki mozaikowej (kody 128—143) są każdorazowo konstruowane. Adres początku tego obszaru —256 jest przechowywany w zmiennej systemowej CHARS.

Ekran (16384 do 22527)

Obszar ten jest przeznaczony do przechowywania danych o każdym punkcie na ekranie telewizyjnym, a dokładniej o tym, czy ma on być w kolorze atramentu czy tła. 6144 bajty tego bloku pozwalają adresować 49152 różne punkty (jeden bajt opisuje 8 kropek). Zorganizowane są one w tablicę o 192 liniach i 256 kolumnach. Dla graficznych instrukcji Basic dostępnych jest jedynie górnych 176 linii. Każdy znak graficzny drukowany jest na polu 8*8 punktów. Pozwala to na pisanie tekstów w formacie 24 wiersze na 32 kolumny. Dwie dolne linie tekstowe są w zasadzie zarezerwowane na komunikaty systemowe i dla obszaru roboczego edytora.

Interesujący jest sposób w jaki obszar ten jest przenoszony na ekran. Niestety kolejne 32 bajty nie opisują kolejnych linii graficznej na ekranie, a kolejnych 256 bajtów nie opisuje wiersza tekstu. Rozmieszczenie linii jest zupełnie odmienne, a z punktu widzenia ZX—Basic wręcz zwariowane. Najłatwiej zaobserwować to w czasie wczytywania ekranu z taśmy. Tutaj ograniczymy się jedynie do podania wzorów pozwalających przeliczyć współrzędne tekstowe i graficzne na adresy ekranowe.

Bajt zawierający punkt graficzny o współrzędnych k,m (punkt 0,0 leży w lewym dolnym rogu ekranu powyżej dwóch najniższych wierszy tekstowych) na ekranie ma adres 16384 + 32*(INT((175 — m)/8) — INT((175 — m)/64)*8 + 8*(175 — m — INT((175 — m)/8)*8) + 64*INT((175 — m)/64) + INT(k/8). Położenie bitu w bajcie oblicza się jako 8 — k + INT(k/8)*8. (Powyższe wzory chyba w pełni uzasadniają uznanie sposobu adresowania za zwariowany).

Z kolei adresy znaków tekstowych wyliczamy ze wzoru 16384 + 2048*INT(k/8) + 32*(k — 8*INT(k/8)) + 256*m + n, gdzie k oznacza numer wiersza (0...23), n jest numerem kolumny (0...31), zaś m jest numerem linii graficznej formującej dany znak (0...7). Jako 0—ową linię przyjmuje się tę położoną na samej górze bloku znaku. Początek układu współrzędnych dla instrukcji piszących znakami jest dla odmiany umieszczony w lewym górnym rogu ekranu.

Atrybuty (22528 do 23295)

W tym obszarze przechowuje się informację o rozmieszczeniu kolorów na ekranie. Najmniejszą jednostką powierzchni, której kolor można modyfikować, jest pole pojedynczego znaku o rozmiarach 8*8 punktów. Dla takiego pola można określić barwę tła, atramentu oraz to, czy znak ma być stabilny, czy też kolory tła i atramentu mają być cyklicznie wymieniane celem wywołania efektu migania danego symbolu na ekranie. Do wyboru mamy osiem barw: 0 — czarny, 1 — ciemnoniebieski, 2 — czerwony, 3 — fioletowy, 4 — zielony, 5 — jasnoniebieski, 6 —żółty i 7 — biały. Kolor tła może ponadto występować w dwóch odcieniach: zwykłym i intensywnym (BRIGHT 0,1).

Wszystkie informacje dotyczące pojedynczego pola znakowego są zakodowane w jednym słowie 8—bitowym w następujący sposób:

znaczenie	miganie znaku	intensywność tła	kolor tła	kolor atramentu
bity	7	6	5 4 3	2 1 0

Aby ustawić pożądane atrybuty pola i ustalić, jaką wartość należy umieścić w odpowiednim bajcie, najwygodniej jest posłużyć się wzorem

128*f + 64*b + 32*p + i gdzie f=0 lub 1 określa miganie (FLASH 0,1), b=0 lub 1 precyzuje odcień tła (BRIGHT 0,1), p jest numerem koloru tła oraz i jest numerem koloru atramentu. Przyporządkowanie adresu bajtu opisującego atrybuty pola znakowego o współrzędnych k,m jest naturalne i łatwo się oblicza ze wzoru:

22528 + 32*k + m.

Bufor drukarki (23296 do 23551)

W tym obszarze zbierane są dane do wysłania na drukarkę. Faktyczny wydruk następuje po zapełnieniu tego bufora (256 bajtów lub 32 znaki), lub po wysłaniu symbolu końca linii (CHR$ 13). Jeśli drukarka nie jest dołączona, to obszar ten nie zostanie wykorzystany przez system i może być użyty do innych celów.

Zmienne systemowe (23552 do 23733)

Temu obszarowi poświęcamy następny rozdział.

Mapa Microdrive'a (23734 do CHANS—1)

Obszar ten jest wykorzystywany jedynie w przypadku dołączenia do Spectrum ZX Interface 1. W "gołym" komputerze mapa ta fizycznie nie istnieje i CHANS = 23734.

Informacje o kanałach (CHANS do PROG — 2)

Zawartość tego bloku jest opisana w rozdziale "Kanały i strumienie".

Program Basic (PROG do VARS — 1)

Po zainicjowaniu systemu, bez dołączonych urządzeń zewnętrznych, blok ten zaczyna się od adresu 23755=#5CCB. Przechowywany w nim jest tekst programu Basic. Poszczególne linie w pamięci komputera mają następującą postać:

2 bajty	2 bajty	długość tekstu	1 bajt
nr linii	długość tekstu + 1	tekst	13 = #0D kod klawisza ENTER

W przypadku numeru linii, wyjątkowo odstąpiono od zasad i jako pierwszy występuje starszy bajt, a dopiero potem młodszy. Wszystkie słowa kluczowe występujące w tekście są kodowane pojedynczymi symbolami.

Interesujący jest sposób przechowywania liczb w tekście programu. Za ciągiem znaków reprezentujących kolejne cyfry umieszcza się zawsze symbol kontrolny CHR$ 14, a za nim pięć bajtów zawierających postać binarną tej liczby. W czasie wydruku programu na ekranie, tych dodatkowych sześć bajtów jest pomijanych. Wyjaśnia to, dlaczego niektórzy programiści preferują zapis VAL "17" zamiast prostego 17. Pierwszy z nich faktycznie zajmuje pięć bajtów, a drugi 8. Chodzi więc o oszczędność pamięci. W czasie wykonywania programu z kolei pomijane są symbole służące do zapisu liczby i korzysta się z 5-bajtowej postaci binarnej. Pozwala to na ukrywanie prawdziwych wartości pewnych liczb (np. adresów startowych procedur w kodzie maszynowym) przed "ciekawskimi". Trzeba w tym celu ustalić adres postaci wewnętrznej danej liczby i instrukcją POKE nadać jej pożądaną wartość. W wydruku tekstu na ekranie w dalszym ciągu będzie figurować stara postać liczby nie związana już z zapisaną za nią liczbą binarną! Stosując tę technikę trzeba pamiętać, że każdorazowe ściągnięcie takiej linii na dół ekranu i odesłanie z powrotem, nawet bez żadnych modyfikacji, zmienia postać wewnętrzną każdej liczby w tekście, nadając jej wartość widoczną na ekranie.

Z innych ciekawych trików wymieńmy jeszcze dwa. Jeśli w pierwszej linii programu, w komórkach zawierających jej numer umieścimy liczbę 0, to wiersza takiego nie da się skopiować do obszaru edytora, a zatem i modyfikować. W drugim końcu programu możliwa jest inna sztuczka. Dopiszmy tam linię 9999 REM i po ustaleniu jej adresu w obu komórkach zawierających jej numer umieśćmy wartość 255. Pierwszy skutek takiego zabiegu objawi się w trakcie drukowania programu na ekranie. Zmodyfikowana przez nas linia nie pojawi się. Drugi efekt, znacznie cenniejszy, da się zaobserwować przy próbie wczytania takiego programu z kasety instrukcją MERGE. Spectrum obraża się na użytkownika i przestaje reagować na jakiekolwiek klawisze.

Zmienne języka Basic (VARS do E_LINE — 2)

W obszarze tym system przechowuje wszystkie zmienne inicjowane zarówno przez program, jak i przez użytkownika z klawiatury. Pojawiające się zmienne są kolejno dopisywane na końcu tego obszaru (wszystkie dalsze bloki są automatycznie przesuwane) i pozostają tam aż do jego wyczyszczenia. Wyjątkiem są proste zmienne tekstowe: gdy którąś z nich jest modyfikowana jej poprzednia wersja jest usuwana (czemu towarzyszą niezbędne przesunięcia innych bloków pamięci), a nowa jest dopisywana na końcu tego obszaru. Podobnie dzieje się przy ponownym deklarowaniu tablic.

Sposób przechowywania zmiennych zależny jest od ich typów. W ZX Basic istnieje sześć różnych rodzajów zmiennych oznaczanych liczbami od 2 do 7:

(010) — proste zmienne tekstowe,
3 — (011) — proste zmienne numeryczne o nazwach jednoliterowych,
4 — (100) — tablice numeryczne,
5 — (101) — proste zmienne numeryczne o nazwach wieloliterowych,
6 — (110) — zmienne sterujące pętlami FOR...NEXT.
(W nawiasach podane są postaci binarne liczb określających typ zmiennej).

We wszystkich przypadkach pierwszy bajt opisu zmiennej zawiera jej typ oraz numer pierwszej litery nazwy. Metoda ujęcia tych informacji w jednym słowie jest następująca:

	typ	nr litery
bity	7 6 5	4 3 2 1 0

Zwróćmy uwagę, że bity 4...0 zawierają numer kolejny litery w alfabecie, a nie jej kod. Właściwy kod uzyskujemy przez dodanie liczby 96 (#60; przechowując nazwy zmiennych, Spectrum automatycznie zastępuje duże litery małymi i pomija wszystkie spacje). Zawartość dalszych bajtów uzależniona jest od typu zmiennej. Przedstawiają to poniższe diagramy:

typ i litera	n=długość tekstu (2 bajty)	tekst

prosta zmienna tekstowa, długość opisu = n + 3 bajty

typ i litera	wartość (5 bajtów)

prosta zmienna numeryczna o nazwie jednoliterowej, długość opisu = 6 bajtów

typ i litera	n=długość opisu—3 (2 bajty)	k=liczba wymiarów	1—szy...k—ty wymiar...wymiar (po 2 bajty)	elementy (po 5 bajtów)

tablica numeryczna, długość opisu = n+3 bajty

typ i litera	2—gi znak ... k—ty znak nazwy ... nazwy	wartość (5 bajtów)

prosta zmienna numeryczna o nazwie wieloznakowej, długość opisu = k+5 bajtów

Drugi i dalsze symbole nazwy są przechowywane już jako kody odpowiednich znaków. Ostatni bajt nazwy ma najstarszy bit ustawiony zawsze na 1, co pozwala rozpoznać go.

typ i litera	n=długość opisu — 3 (2 bajty)	k=liczba wymiarów	1—szy ... k—ty wymiar ... wymiar (po 2 bajty)	elementy (po 1 bajcie)

tablica znakowa, długość opisu = n+3 bajty

typ i litera	wartość chwilowa=x 5 bajtów)	wartość końcowa=y 5 bajtów)	krok=z (5 bajtów)	k=nr linii (2 bajty)	nr FOR w linii+1 (1 bajt)

zmienna sterująca (k...:FOR α = x TO y STEP z:...), długość opisu = 19 bajtów

Ostatni bajt obszaru zmiennych jako znacznik zawsze zawiera wartość 128= #80.

Bufor edytora (E_LINE do WORKSP — 1)

W tym bloku umieszcza się linię programu lub ciąg rozkazów do natychmiastowego wykonania podczas wprowadzania ich z klawiatury. Wprowadzane symbole są kopiowane na dół ekranu. Po naciśnięciu ENTER komputer sprawdza poprawność wprowadzonego tekstu i podejmuje odpowiednie dalsze działania.

Bufor instrukcji INPUT... (WORKSP do STKBOT — 1)

Do tego bufora są wstępnie wprowadzane dane wczytywane rozkazem INPUT i dopiero po wciśnięciu ENTER następuje ich przetwarzanie. Również w tym obszarze Spectrum dokonuje niektórych operacji wymagających pamięci roboczej (np. łączenie łańcuchów znaków).

Stos kalkulatora (STKBOT do STKEND — 1)

Obszar roboczy zestawu podprogramów systemowych wykonujących większość operacji arytmetycznych na zmiennych języka Basic.

Obszar wolny systemu Basic (STKEND + 1 do SP)

Ten blok pamięci jest wolny, ale przeznaczony na potrzeby systemu Basic. Obszary robocze z obu końców tego bloku zmieniają swoją objętość jego kosztem. Umieszczone tu dane mogą ulec zniszczeniu bez żadnego uprzedzenia. Przed podjęciem działań wymagających powiększenia któregokolwiek z obszarów roboczych, Spectrum sprawdza czy w bloku zostanie jeszcze co najmniej 80 wolnych bajtów na ewentualne potrzeby stosu maszynowego. Negatywny wynik testu jest sygnalizowany komunikatem 4. Adres końca tego obszaru jest przechowywany w rejestrze procesora Z80, zwanym wskaźnikiem stosu (SP) i jest niedostępny z systemu Basic.

Stos maszynowy (SP do ERR_SP)

Stos używany przez procesor Z80 do chwilowego przechowywania różnych informacji, adresów itp. Jest to obszar całkowicie bezużyteczny dla programujących w języku Basic.

Stos adresów powrotnych GO SUB (ERR_SP + 1 do RAMTOP)

Kolejny stos, tym razem przeznaczony do przechowywania numeru linii i położenia w niej instrukcji GO SUB. Dane te są niezbędne dla rozkazu RETURN do właściwego powrotu z podprogramu. Każdorazowe wykonanie GO SUB lub RETURN powoduje przesunięcie stosu maszynowego adresowanego przez ERR SP o trzy bajty w dół lub w górę. Z tego względu przy modyfikowaniu bloku wskazana jest szczególna ostrożność.

Obszar wolnej pamięci (RAMTOP + 1 do P_RAMT)

Blok pamięci rzeczywiście wolnej i bezpiecznej w tym sensie, że jego zawartość jest niedostępna dla systemu Basic i modyfikowana może być jedynie na wyraźne życzenie użytkownika za pomocą instrukcji POKE. Nawet rozkaz NEW nie narusza tego obszaru. Z reguły jest on wykorzystywany do przechowywania programów w kodzie maszynowym lub nietypowych danych, dla których zmienne Basic niezbyt się nadają.

Wzory symboli użytkownika (UDG do UDG + 167)

Po zainicjowaniu systemu, blok ten rozpoczyna się w komórce o adresie RAMTOP + 1 = 65368 = #FF58 i kończy się w P RAMT = 65535 = #FFFF. Adres tego bloku w pamięci można uzyskać wykonując instrukcję USR "a". Obszar ten służy do przechowywania kształtów znaków graficznych zdefiniowanych przez użytkownika. Definicja pojedynczego symbolu składa się z ośmiu bajtów opisujących ustawienie każdej kropki w polu znakowym o rozmiarach 8*8.

Dla przykładu zdefiniujmy wzór małej polskiej litery "ó" i umieśćmy go w obszarze UDG tak, by był wyświetlany po naciśnięciu klawisza "O" w trybie G. Pierwszym etapem jest zaprojektowanie kształtu nowego znaku w kwadracie 8*8. W polu przeznaczonym do zamazania umieszczamy 1, a w pustym 0. Każdy tak zapełniony wiersz traktujemy jako ośmiocyfrową liczbę binarną. Otrzymany zestaw ośmiu liczb opisuje kształt projektowanego właśnie symbolu.

```
0 0 0 0 0 1 0 0   4
0 0 0 0 1 0 0 0   8
0 0 0 0 0 0 0 0   0
0 0 1 1 1 1 0 0  60
0 1 0 0 0 0 1 0  66
0 1 0 0 0 0 1 0  66
0 0 1 1 1 1 0 0  60
0 0 0 0 0 0 0 0   0
```

Wprowadźmy go teraz do komputera. Najprościej jest to zrobić za pomocą poniższego programu:

```
10 DATA 4,8,0,60,66,66,60,0
20 FOR I=USR "O" TU USR "O" + 7
30 READ A : POKE I,A
40 NEXT I
```

Po jego wykonaniu klawisz "O" w trybie G drukuje literę "ó". W łańcuchach znaków będzie ona miała kod 158. Dane w linii 10 można zamiast w postaci dziesiętnej podać w postaci binarnej korzystając z funkcji BIN. Wymaga to dłuższego pisania, ale za to znacznie ułatwia wszelkie modyfikacje definiowanego kształtu.

168 bajtów obszaru UDG pozwala na przechowywanie 21 różnych nowych symboli. Ich kody mieszczą się w zakresie od 144 (A) do 164 (U). Po zainicjowaniu systemu, blok ten zawiera wzory kształtów dużych liter alfabetu łacińskiego.

Rozdział VI

ZMIENNE SYSTEMOWE

Zmienne systemowe przechowywane są począwszy od adresu 23552 = #5C00 aż do obszaru zajmowanego przez program Basic. Programy z pamięci ROM wykorzystują je do zapamiętania informacji opisujących stan komputera. Poniżej omawiamy te zmienne, które mogą być modyfikowane i z pożytkiem wykorzystane przez programistę. Pozostałych lepiej nie ruszać, bo może to doprowadzić do załamania się systemu operacyjnego. Opisywane zmienne podzielimy (według spełnianych funkcji, a nie położenia w pamięci) na pięć grup:

1. przechowujące ważne dla systemu adresy
2. obsługujące klawiaturę,
3. opisujące stan systemu,
4. związane z kanałami i strumieniami,
5. obsługujące ekran telewizora.

Do **pierwszej grupy** zaliczamy:

ERR SP	23613 = #5C3D
VARS	23627 = #5C4B
PROG	23635 = #5C53
E_LINE	23641 = #5C59
WORKSP	23649 = #5C61
STKBOT	23651 = #5C63
STKEND	23653 = #5C65
RAMTOP	23730 = #5CB2
P_RAMT	23732 = #5CB4
CHARS	23606 = #5C36
UDG	23675 = #5C7B

Podstawowe znaczenia tych zmiennych opisaliśmy w poprzednim rozdziale. Naturalnym zastosowaniem tych danych może być szacowanie rozmiarów pamięci wolnej lub zajętej przez program, zmienne itd. Jedynie zmienna RAMTOP może być łatwo modyfikowana z języka Basic za pośrednictwem CLEAR k. Niestety rozkaz ten ma również inne działania uboczne, co czasem uniemożliwia jego stosowanie.

P_RAMT zawiera adres ostatniej, fizycznie obecnej w komputerze, komórki pamięci. Może być więc wykorzystywana przez program do rozpoznawania w jakim modelu komputera Spectrum jest załadowany (16K lub 48K).

CHARS. Dzięki tej zmiennej możemy definiować własne kroje pisma. Po umieszczeniu ich w pamięci RAM pod adresem k wystarczy nadać zmiennej CHARS wartość k — 256. Począwszy od tego momentu wszystkie teksty, łącznie z listingiem programów, będą drukowane naszymi nowymi znakami. Zmienić możemy w ten sposób kształty wszystkich symboli o kodach od 32 do 127. Warto zaznaczyć, że modyfikujemy tak tylko kształty znaków, a nie ich znaczenia. Te zależą od kodów symboli i jako takie pozostają bez zmian. Można użyć tej zmiennej do trików czyniących tekst programu całkowicie nieczytelnym. Wystarczy zmniejszyć wartość CHARS o 8, a już w wydruku programu ciężko będzie doszukać się jakiegokolwiek sensu. Natomiast napisy wykonywane przez uruchomiony program pozostaną czytelne, jeśli będą zawczasu przesunięte o jedną wartość kodu, tzn. w pierwotnym tekście umieścimy "B" zamiast "A", "o" zamiast "n" itd.

UDG. Wygoda z umieszczenia tego adresu na liście zmiennych systemowych polega głównie na tym, że blok ten możemy w pamięci dowolnie przemieszczać. Równie cenna jest możliwość korzystania z kilku zestawów nietypowych symboli jednocześnie. W takim przypadku przed drukowaniem danego znaku trzeba modyfikować UDG nadając zmiennej wartość potrzebnego w danej chwili zestawu.

Modyfikowanie pozostałych zmiennych tej grupy jest bardzo niebezpieczne i łatwo może doprowadzić do załamania się systemu operacyjnego.

Osobno warto wspomnieć o **ERR_SP**. Zmienna ta wskazuje dno stosu maszynowego. Tam z kolei jest umieszczony adres procedury w pamięci ROM, odpowiedzialnej za obsługę wszelkich sytuacji błędnych, wymagających przerwania programu i wydruku odpowiedniego komunikatu. Przez błąd rozumie się tu więc również poprawne zakończenie programu sygnalizowane komunikatem 0 OK.

Zaawansowani programiści znający asembler Z80 mogą dzięki temu pisać własne procedury obsługi błędów, co pozwala m.in. na dodawanie nowych komend do języka. Modyfikowanie ERR_SP z poziomu systemu Basic na ogół prowadzi do załamania systemu w chwili powstania sytuacji wymagającej napisania jakiegokolwiek komunikatu. Bywa to często wykorzystywane do ochrony programu przed nieautoryzowanym listowaniem i kopiowaniem. Metoda ta połączona z nagrywaniem programów przez SAVE f LINE... oraz z wyeliminowaniem możliwosci wczytania programu instrukcją MERGE (patrz rozdział "Wykorzystywanie pamięci" punkt "Program Basic") zazwyczaj w zupełności wystarcza do zabezpieczenia się przed mniej zaawansowanymi piratami. Niestety nie chroni ona przed licznymi programami kopiującymi.

Druga grupa zmiennych zajmuje się obsługą klawiatury.

KSTATE 23552 = #5C00

Zmienna ta, zbudowana z ośmiu bajtów, jest wykorzystywana do czytania klawiatury i obsługi samopowtarzalności klawiszy. Dla programisty jedynie komórka o adresie 23556 = #5C04 może mieć znaczenie. Zawiera ona wartość 255, jeśli żaden klawisz nie jest wciśnięty, lub kod głównego znaczenia wciśniętego klawisza w trybie C (duża litera alfabetu lub cyfra). PEEK 23556 daje zatem ten sam efekt co CODE INKEY$ w trybie C. Zaletą jest fakt, że jej zawartość nie zależy od stanu kursora (L lub C). W przypadku wciśnięcia kilku klawiszy jednocześnie zawsze zostanie rozpoznany pierwszy (INKEY$ w takiej sytuacji może zignorować wszystkie). Klawisze CS i SS naciskane oddzielnie nie mają wpływu na naszą komórkę, ale wciśnięte razem dają kod 14. Ponadto,kombinacje CS/9, CS/3, CS/4 i CS/2 produkują odpowiednio kody 15, 4, 5, 6. (Kody te wyłapuje również CODE INKEY$).

LAST_K 23560 = #5C08

Ta jednobajtowa zmienna przechowuje kod ostatnio wciśniętego klawisza niezależnie od tego, czy jest on przytrzymywany, czy już nie. W razie wciśnięcia kilku — zapamiętany będzie jedynie pierwszy. W odróżnieniu od poprzedniej zmiennej, LAST K uwzględnia stan kursora. Ze zmienną tą częściowo jest związana zmienna

FLAGS 23611 = #5C3B

Ilekroć zmienna LAST K przyjmuje nową wartość piąty bit zmiennej FLAGS jest ustawiany na 1. Jest to ważne, gdy chcemy wykryć wielokrotne wciskanie tego samego klawisza.

Z kolei trzeci bit tej zmiennej, wraz z zawartością zmiennej MODE, pozwala rozpoznać w jakiej sytuacji do czytania klawiatury ma być użyty kursor K. Tę informację można jednak wykorzystać dopiero z poziomu asemblera. Najstarszy bit FLAGS (siódmy) sygnalizuje systemowi, czy rozkaz wykonywany jest z programu czy też z klawiatury. Umieszczona w programie instrukcja POKE 23611,0 przerwie jego wykonywanie zazwyczaj komunikatem 0 OK.

REPDEL 23561 = #5C09
REPPER 23562 = #5C0A

Te dwie zmienne należy rozpatrywać razem. Pierwsza z nich określa, jak długo należy przyciskać klawisz, by uruchomić mechanizm samopowtarzania, a druga określa czas między powtórzeniami odczytu klawisza. W obu zmiennych czas jest podany w 1/50 sekundy. Początkowo zmienne te są inicjowane wartościami 35 i 5 co oznacza, że wciśnięty klawisz po 0.7 s zacznie się powielać w tempie 10 znaków na sekundę. W trakcie czytania klawiatury wartości te są zmniejszane aż do osiągnięcia zera. Nadając im wartości 0 osiąga się największe opóźnienie, trwające tyle samo co PAUSE 256. Umieszczenie w tych komórkach wartości 1 sprawia, że współpraca ze Spectrum staje się trudna, ale nie niemożliwa. (Wprowadzanie jakiegokolwiek rozkazu z klawiatury będzie wymagało od operatora sporego refleksu i wyczucia w palcach!).

RASP 23608 = #5C38
PIP 23609 = #5C39

Te zmienne określają dźwięki towarzyszące wprowadzaniu danych. RASP określa jak długo ma trwać (jednostką jest 1/50 sek) ostrzegawcze buczenie, gdy Spectrum nie chce więcej czytać danych (np. przy wprowadzaniu jednorazowo linii nie mieszczącej się w 22 wierszach na ekranie). Początkowo RASP=64 (1,26 sek).

PIP określa czas trwania dźwięku potwierdzającego naciśnięcie klawisza (początkowo PIP=0).

Warto pamiętać, że RASP i PIP nie są, w odróżnieniu od większości pozostałych zmiennych systemowych, ponownie inicjowane przez instrukcję NEW.

Do **trzeciej grupy** zaliczamy zmienne, które pozwalają systemowi kontrolować sytuację i poprawnie interpretować kolejne linie programu:

NEWPPC 23618 = #5C42
NSPPC 23620 = #5C44

Pierwsza z nich (dwubajtowa) zawiera numer linii, a druga (jednobajtowa) numer rozkazu w linii, do którego ma nastąpić skok. Rozkazy
POKE 23618, L—256*(INT(L/256)):
POKE 23619, INT(L/256):
POKE 23620, m powodują natychmiastowy skok do m—tej instrukcji w L—tej linii, działają więc znacznie precyzyjniej od rozkazu GO TO L.

PPC 23621 = #5C45
SUBPPC 23623 = #5C47

Zmienne te zawierają odpowiednio numer linii i numer wykonywanego rozkazu. Razem ze zmienną ERR_NR mogą posłużyć do kończenia programu z wcześniej zadanym komunikatem bez prowokowania faktycznego błędu. W tym celu program należy zakończyć instrukcją GO TO 9999, a w tej linii umieścić rozkazy:
9999 POKE 23621, L — 256*INT (L/256):
POKE 23622, INT (L/256):
POKE 23610, m—1:POKE 23623,k Program zakończy się wtedy komunikatem m tekst L : k

ERR_NR 23610 = #5C3A

W tej komórce pojawia się numer — 1 błędu, który wystąpił i będzie właśnie sygnalizowany odpowiednim komunikatem. Zawartość tego bajtu może być cenna przy pisaniu własnych procedur obsługi błędów w asemblerze, gdyż od razu wyjaśnia przyczynę przerwania programu.

E_PPC 23625 = #5C49

Jest to dwubajtowa zmienna zawierająca numer linii bieżącej, a więc dostępnej w danym momencie do edycji. Jej modyfikowanie wywołuje zatem ten sam efekt co LIST n, ale bez wydruku programu na ekranie. Umożliwia to wybieranie z klawiatury linii programu do modyfikacji, a następnie skopiowanie jej na dół ekranu i poprawianie bez niszczenia zawartości ekranu.

MODE 23617 = #5C41

Zmienną tę wykorzystuje edytor do określenia, jaki kursor ma być w danym momencie użyty. System wykorzystuje następujące wartości: 0 — L,C lub K,1—E oraz 2—G. Liczby te są następnie używane w różnych wyrażeniach do wyznaczania kodu symbolu, który ma być wyświetlany jako kursor. Programując w języku Basic można jedynie nakłonić Spectrum, by w najbliższej instrukcji INPUT zamiast standardowego kursora L lub C, użył G lub E, przy czym tryb E będzie skuteczny tylko dla pierwszego wprowadzanego symbolu. Tryb E wymusza POKE 23617,1. Interesujące efekty daje umieszczenie w tej zmiennej innych wartości niż podane wyżej. POKE 23617,k dla k=2,3.....,127 wymusza kursor G, choć na ogół na ekranie nie będzie drukowana litera G, a coś całkiem innego. Podobnie POKE 23617,k dla k=128,...,255 oraz 0 ustanawia tryb L lub C, ale z drukowaniem na ekranie innego migającego kursora. Zachęcamy Czytelników do eksperymentowania (naszym zdaniem parzyste wartości k są lepsze).

FLAGS2 23658 = #5C6A

Ta zmienna określa, czy obowiązuje tryb L czy C. Odpowiada za to trzeci bit FLAGS2. POKE 23658,0 ustawia tryb L a POKE 23658,8 tryb C. Podobne zmiany uzyskuje się z klawiatury przez CS/2.

DF_SZ 23659 = #5C6B

Zmienna ta zawiera liczbę linii dolnej części ekranu, zarezerwowanej na komunikaty systemowe i dane wprowadzane przez INPUT. Inicjowana jest na 2. W zasadzie jest bezużyteczna, choć bywa niekiedy stosowana do ochrony programów przed przerwaniami. Umieszczenie w niej wartości 0 powoduje zawieszenie się systemu, ilekroć zaistnieje potrzeba drukowania czegokolwiek w dolnej części ekranu (np. komunikatu o wciśnięciu BREAK). Nie jest to jednak wygodna broń, gdyż uniemożliwia jednoczesne stosowanie instrukcji INPUT, CLS oraz nie pozwala na zadanie pytania "scroll?".

OLDPPC 23662 = #5C6E
OCPPC 23664 = #5C70

Zawierają one numer linii i rozkazu w linii, do których nastąpi skok w razie wydania rozkazu CONTINUE. Modyfikacje tych zmiennych w programie pozwalają na uzyskanie precyzyjniejszej odmiany instrukcji GO TO. W niektórych sytuacjach jest to wygodniejsze niż modyfikowanie NEWPPC i NSPPC, gdyż skok następuje dopiero po natrafieniu na CONTINUE, a nie natychmiast po zmodyfikowaniu zmiennych. Zmienne te są automatycznie korygowane przez system, ilekroć nastąpi przerwanie programu z komunikatem różnym od 0 OK.

SEED 23670 = #5C76

Jest to "ziarno„ generatora pseudolosowego. Wykorzystanie SEED omówiliśmy przy okazji funkcji RND.

FRAMES 23672 = #5C78

Trzy bajty tworzą wewnętrzny zegar ZX Spectrum. Przedstawiają one liczbę PEEK 23672 + 256*PEEK 23673 + 65536*PEEK 23674

i określają, ile 1/50 sekundy upłynęło od inicjacji systemu. Maksymalną wartością jest więc 2 ↑ 24 — 1 = 16777215, co odpowiada 3 dobom, 21 godzinom, 12 minutom i 24,3 sekundy. Dokładność zegara wynosi ok. 0.01%, czyli mniej więcej 9 sekund na dobę. Zegar ten jest wyłączany na czas, gdy Spectrum obsługuje urządzenia zewnętrzne (takie jak drukarka, magnetofon, głośnik) oraz w czasie wykonywania programów w kodzie maszynowym, które wyłączają lub przejmują kontrolę tzw. przerwań maskowalnych.

DATADD 23639 = #5C57

Ta zmienna przechowuje adres elementu na liście DATA, który będzie wczytany następną instrukcją READ (jest to tylko bliskie prawdy, ale wystarczająco dobre przybliżenie faktycznej zawartości tych dwóch komórek). Zapamiętując jej zawartość i odtwarzając ją później można uzyskać precyzyjniejszą wersję rozkazu RESTORE k.

SCR_CT 23692 = #5C8C

Bajt ten określa, po wydrukowaniu ilu linii + 1, na ekranie ma paść pytanie "scroll?". Jeśli chcemy, by wydruk nie był przerwany, to przynajmniej raz na 255 drukowanych linii powinniśmy umieścić w tej komórce wartość 255. Podobnie, jeśli zależy nam na przerywaniu wydruku wcześniej (nie chcemy tracić całej zawartości ekranu), to musimy nadać tej zmiennej wartość mniejszą od 23.

Zmienne związane z kanałami i strumieniami omówione zostaną w następnym rozdziale.

Ostatnia grupa zmiennych odpowiedzialna jest za prawidłową współpracę komputera z telewizorem. Jedne z nich sterują kolorami, inne zaś określają miejsce, w którym ma być wyświetlany kolejny znak lub punkt graficzny.

BORDCR 23624 = #5C48

Zmienna ta zawiera atrybuty opisujące dolną część ekranu oraz kolor ramki. W normalnych warunkach Spectrum nie pozwala na ustawienie jednakowej barwy tła i atramentu w dolnych liniach (użytkownik zawsze powinien widzieć wprowadzane przez siebie symbole). Specyfikatory kolorów na liście INPUT będą skuteczne jedynie dla tekstów drukowanych w dole ekranu, ale kolor atramentu dla wprowadzanych danych jest zawsze ustawiany na 9 (biały lub czarny zależnie od koloru tła). Ominąć to można (np. jeśli komputer ma wczytywać z klawiatury tajne hasło) stosując rozkaz: POKE 23624,128*f + 64*b + 8*p + i nadając literom f, b, p, i wartości parametrów rozkazów FLASH, BRIGHT, PAPER i INK, niezbędnych do uzyskania pożądanego efektu. Nieprzekonanym użytkownikom proponujemy uzyskanie w inny sposób efektu wywoływanego przez rozkazy: BORDER 3:POKE 23624,222:CLS.

ATTR_P 23693 = #5C8D
ATTR_T 23695 = #5C8F

Obie zmienne są jednobajtowe i przechowują wartości atrybutów FLASH, BRIGHT, PAPER oraz INK. Litera P oznacza wartości stałe ustanowione przez odpowiednie rozkazy dla całego programu, zaś T wartości tymczasowe ustanawiane przez te same instrukcje umieszczone na listach odpowiednich komend piszących lub rysujących. Przy braku specyfikatorów barw na listach instrukcji piszących, zmienna ATTR_P jest kopiowana do ATTR_T. Sposób zawarcia informacji o kolorach w jednym bajcie opisaliśmy w rozdziale "Wykorzystanie pamięci" w dziale dotyczącym atrybutów. Zmienna ATTR_P może być wykorzystana w programie Basic do ustawiania wszystkich atrybutów jedną instrukcją POKE. ATTR_T z kolei będzie ważna dla programujących w kodzie maszynowym, gdyż jest zazwyczaj używana przez procedury w pamięci ROM do ustalenia obowiązujących barw.

MASK_P 23694 = #5C8E
MASK_T 23696 = #5C90

Zmienne te (jednobajtowe) mają zastosowanie przy realizacji rozkazów FLASH, BRIGHT, PAPER oraz INK z parametrem 8. Znaczenie liter P i T w nazwach zmiennych jest takie samo, jak dla zmiennych ATTR. Ustawienie któregokolwiek bitu tych zmiennych na 1 oznacza, że bit o tym samym numerze w odpowiednim bajcie atrybutów ma pozostać niezmieniony. Zauważmy, że rozkaz INK 8 ustawia na 1 wszystkie trzy najmłodsze bity zmiennej MASK P. Modyfikując tę zmienną instrukcją POKE możemy np. ustawić na jeden tylko najmłodszy bit. Uzyskamy wtedy efekt filtru. Niezmieniona w kolorze atramentu zostanie jedynie barwa podstawowa niebieska, podczas gdy pozostałe mogą ulec zmianie (zwróćmy uwagę, że numeracja kolorów nie jest wcale przypadkowa. Barwy niebieska—1, czerwona—2, zielona—4 to kolory podstawowe, których mieszanie pozwala uzyskiwać wszystkie inne. W Spectrum role mieszania pełni dodawanie numerów odpowiednich kolorów!).

COORDS 23677 = #5C7D

Dwa kolejne bajty tej zmiennej zawierają współrzędne x i y punktu na ekranie, w którym zakończyła rysowanie ostatnia instrukcja PLOT, DRAW lub CIRCLE. Modyfikowanie tej zmiennej daje ten sam efekt, co PLOT OVER 1,k,n:PLOT OVER 1,k,n, a więc przesunięcie wskaźnika ekranowego bez rysowania żadnego punktu ani linii na ekranie.

SPOSN 23688 = #5C88

Zmienna ta w kolejnych dwóch bajtach zawiera wartości 33 — k oraz 24 — m, gdzie k,m są współrzędnymi ostatnio wyświetlonego na ekranie znaku. Bezpośrednie modyfikowanie tych zmiennych jest utrudnione, gdyż trzeba jednocześnie modyfikować DF_CC.

DF_CC 23684 = #5C84

Zmienna ta zawiera adres bajtu na ekranie, od którego zacznie się drukowanie następnego symbolu przez instrukcję PRINT. Modyfikowana powinna być razem z S_POSN. Znacznie prostsze jest użycie rozkazu ZX—Basic PRINT AT k,n;.

SPONSL 23690 = #5C8A
DFCCL 23686 = #5C86

Zmienne analogiczne do S_POSN i DF_CC, opisujące dolną część ekranu.

P_FLAG 23697 = #5C91

Zmienna systemowa zawierająca informacje o trybie drukowania i rysowania na ekranie. Opisuje tryby ustawiane przez język Basic instrukcjami INVERSE, OVER, INK 9 oraz PAPER 9. Bity nieparzyste odpowiadają ustawieniu tych trybów stale, parzyste zaś tymczasowo:

	tymczasowo bit	stale bit
OVER 1	0	1
INVERSE 1	2	3
INK 9	4	5
PAPER 9	6	7

Jeśli chcemy na stałe ustawić OVER 1 : INVERSE 1 : INK 9 : PAPER 9, to zamiast tych czterech rozkazów wystarczy jeden: POKE 23697,2 + 8 + 32 + 128.

UWAGA: Nazwy zmiennych systemowych nie są rozpoznawane przez system Basic. Pochodzą one od autorów systemu operacyjnego i są powszechnie przyjęte w literaturze dotyczącej Spectrum.

Rozdział VII

KANAŁY I STRUMIENIE

Przepływ informacji między programem użytkownika i urządzeniami zewnętrznymi sterowany jest przez kanały i strumienie. **Wygodnie jest wyobrazić sobie kanał jako fizyczne urządzenie odbierające informacje (telewizor, drukarka), lub wysyłające ją (klawiatura), strumień zaś jako ścieżkę, po której dane mają płynąć do lub z kanału.**

W ZX Spectrum bez ZX Interface 1 rozpoznawane są cztery kanały. Oznacza się je pojedynczymi literami (dużymi lub małymi):

"S" — kanał wyjściowy, dane przesyłane tym kanałem są wyświetlane w górnej części ekranu telewizora
"K" — kanał wejścia/wyjścia, obsługuje klawiaturę i dolną część ekranu telewizora
"P" — kanał wyjściowy, dane są wysyłane na drukarkę
"R" — kanał wyjściowy używany jedynie przez edytor do wpisywania do bufora edytora danych wczytanych z klawiatury.

Dane do kanałów przesyła się dołączanymi do nich strumieniami. Mamy ich do dyspozycji szesnaście numerowanych liczbami naturalnymi z przedziału 0...15. Po zainicjowaniu systemu następuje przyłączenie strumieni 0 i 1 do kanału "K", strumienia 2 do "S" oraz strumienia 3 do "P". Kanał "R" jest niedostępny z ZX—Basic.

Przewidziane w tym języku instrukcje PRINT, LPRINT, INPUT, LIST oraz LLIST umożliwiają dostęp do wszystkich tych kanałów, skutecznie ukrywając ich istnienie. W rzeczywistości bowiem niezbędne są tylko dwie z nich — INPUT i LIST oraz istniejący system kanałów i strumieni. Każda z tych instrukcji może być użyta do wysyłania informacji dowolnym strumieniem. Trzeba w tym celu jedynie symbolem #k zasygnalizować, z którego kanału chcemy korzystać.

Instrukcja LLIST jest równoważna z LIST #3, a LIST to to samo, co LLIST #2. Można również pisać LIST #0 lub LLIST #0, kierując wydruk do dolnej części ekranu, ale nie jest to zbyt praktyczne, bo system nie pozwala na drukowanie tam więcej niż 22 wierszy ekranowych i sam często czyści ten obszar. Podobnie rozkaz PRINT #1; "To jest dół ekranu" spowoduje wydruk tekstu w dole ekranu tak jak INPUT, natomiast PRINT #3; "O! Masz nawet drukarkę" prześle odpowiedni tekst na drukarkę. Samo PRINT jest równoważne z PRINT #2;.

Czasem może być wygodne użycie INPUT #2; "Podaj swoje imię";#0;A$, co spowoduje wydruk polecenia nie w dole ekranu, a na górze. #0; przed wczytaniem A$ jest konieczne, bo kanał "S" jest jedynie kanałem wyjściowym i nie można z niego czytać.

Do dyspozycji użytkownika pozostają strumienie o numerach od 4 do 15. Przed ich wykorzystaniem trzeba odpowiednie strumienie dołączyć do kanałów, które mają je obsługiwać. Służy do tego instrukcja OPEN #k;"L", gdzie #k jest wyrażeniem dającym w wyniku liczbę między 4 i 15, a L jest wyrażeniem alfanumerycznym, dającym w wyniku jedną literę identyfikującą kanał (w "gołym" Spectrum musi to być K, S lub P). Na przykład po rozkazie OPEN #6;"P" instrukcja LIST #6 będzie działała tak, jak LLIST. Podobnie PRINT #6; może być wykorzystywane zamiast LPRINT.

Po wykorzystaniu danego strumienia przed przyłączeniem do innego kanału konieczne jest jego odłączenie. Służy do tego rozkaz CLOSE #k. Dobry zwyczaj nakazuje zamykać kanały (odłączać od nich strumienie) natychmiast ich po wykorzystaniu. Może to być ważne dla urządzeń zewnętrznych (pozwoli im np. wyłączyć kontrolowane przez siebie urządzenia). Strumienie 0...3 są automatycznie przyłączane do swoich kanałów i ich zamykanie jest niecelowe, gdyż system sam przyłączy je z powrotem. Próby dołączania ich do innych kanałów mogą mieć nieokreślone skutki, do załamania systemu włącznie. Również nie wolno zamykać strumieni, które nie są dołączone do jakiegoś kanału (patrz rozdział "Błędy w systemie").

"Gołe" ZX Spectrum nie daje wielu możliwości pełnego docenienia zalet takiego systemu przepływu informacji (szczególnie gdy nie zachodzi potrzeba współpracy z poziomu ZX—Basic z nietypowymi urządzeniami zewnętrznymi). Ujawniają się one dopiero po przyłączeniu ZX Interface 1, przy pracy z Microdrive'm lub w sieci komputerowej.

Najpoważniejszą korzyścią przyjęcia systemu kanałów i strumieni jest jego duża elastyczność, wyrażająca się łatwością kreowania nowych kanałów. W praktyce pozwala to na znaczne uproszczenie konstrukcji interface'ów, przez umieszczenie w pamięci komputera niezbędnych programów obsługujących dane urządzenie dodatkowe (co również zazwyczaj zmniejsza koszty). Włączanie własnych kanałów do systemu Basic wymaga wtedy jedynie dwóch, zazwyczaj bardzo prostych i krótkich, dodatkowych procedur w kodzie maszynowym. Co prawda teoretycznie do tego celu mogłyby wystarczyć instrukcje IN i OUT, ale w przypadku urządzeń wymagających dopływu sygnałów w odstępie kilku czy kilkunastu mikrosekund pozostaje to tylko teorią.

Dla korzystających z danego urządzenia nie bez znaczenia jest też wygoda pisania programów. Prościej bowiem jest używać PRINT#k lub INPUT#k, niż każdorazowo dopisywać w ZX—Basic procedury przygotowujące dane, a następnie wysyłające je po jednym znaku.

Zanim wyjaśnimy sposoby budowania własnych kanałów musimy wiedzieć, jak Spectrum przechowuje odpowiednie informacje o nich i o strumieniach oraz jak z nich korzysta.

W obszarze pamięci rozpoczynającym się od adresu przechowywanego przez zmienną systemową

CHANS (23631 = #5C4F)

do PROG — 1 umieszczone są podstawowe dane o kanałach. Mają one standardowy format. Opis każdego kanału zajmuje pięć bajtów i ma postać:

adres	rozmiar	znaczenie
x	2 bajty	adres procedury wyjściowej kanału
x+2	2 bajty	adres procedury wejściowej kanału
x+4	1 bajt	kod litery identyfikującej kanał

Procedura wyjściowa będzie wywoływana z kodem kolejnego symbolu w rejestrze A. Procedura wejściowa, by mogła bez przeszkód współpracować z ZX—Basic, powinna dostarczać kolejne kody znaków rozpoznawalnych przez system, sygnalizując dostępność danych ustawieniem znacznika C. Brak danych wejściowych powinien być sygnalizowany zerowaniem znaczników C (CARRY — przeniesienie) i Z (ZERO — zero).

Często zdarza się, że dane urządzenie jest jednokierunkowe (drukarka to tylko urządzenie wyjściowe, a klawiatura — wejściowe). Wówczas, jako adres procedury obsługi niemożliwej operacji podaje się położenie (zazwyczaj w pamięci ROM) procedury postaci

RST 0008

DEFB kod odpowiedniego komunikatu błędu (zawartość komórki następującej po RST 8 jest pobierana jako parametr)

Po zainicjowaniu systemu obszar informacji o kanałach zajmuje 20 bajtów plus jeden zawierający znacznik końca obszaru (128=#80). Zawierają one kolejno:

adres	zawartość
CHANS	adres procedury piszącej w dolnej części ekranu
+2	adres procedury czytającej dane z klawiatury
+4	"K" identyfikator kanału
+5	adres procedury piszącej w górnej części ekranu
+7	adres procedury sygnalizującej błąd
+9	"S" identyfikator kanału
+10	adres procedury wprowadzającej wczytane dane do bufora edytora
+12	adres procedury sygnalizującej błąd
+14	"R" identyfikator kanału
+15	adres procedury obsługi drukarki
+17	adres procedury sygnalizującej błąd
+19	"P" identyfikator kanału
+20	128 = #80 znacznik końca obszaru
+21	pierwszy bajt obszaru PROG — VARS

Jak widać w obszarze tym nie ma miejsca na umieszczenie danych o nowych kanałach. Z istniejących do modyfikacji nadawać się może jedynie "P", gdyż pozostałe są automatycznie odtwarzane przez system. Nie jest to jednak sposób najlepszy, bo pozwala na określenie tylko jednego dodatkowego kanału oraz uniemożliwia jednoczesne wykorzystywanie drukarki. Wygodniej jest przesunąć cały blok PROG — 1 do STEND o odpowiednią ilość bajtów wraz z modyfikacją zmiennych systemowych. Najprościej robi się to procedurą systemową MAKE_ROOM (patrz następny rozdział). Mniej eleganckie, ale równie skuteczne, jest umieszczenie informacji o nowych kanałach gdziekolwiek w bezpiecznym obszarze pamięci RAM.

Informacje, które strumienie przyłączone są do jakich kanałów, mieszczą się w obszarze zmiennych systemowych na 38 bajtach począwszy od

STRMS 23568 = #5C10

Na każdy strumień przeznaczone są dwa bajty. Adres pięciu bajtów opisujący dany kanał ma postać CHANS + x — 1, gdzie x jest zawartością dwóch bajtów związanych z danym strumieniem. Opis strumienia o numerze k mieści się pod adresem STRMS + 6 + 2*k.

W momencie przyłączania strumienia do kanału powyższym komórkom nadaje się odpowiednie wartości. Ilekroć w programie pojawi się instrukcja INPUT #k; lub PRINT #k; wówczas na podstawie danych z tablicy STRMS wyznacza się adres odpowiedniej procedury i umieszcza go w zmiennej systemowej

CURCHL 23633 = #5C51.

Dalej w przypadku instrukcji piszącej ładuje się kolejne symbole do akumulatora i woła tę procedurę. W przypadku czytania odbiera się z akumulatora kolejne znaki, jeśli są one dostępne (ustawiony znacznik C). Strumienie nieaktywne oznaczane są zerem w odpowiednim miejscu tablicy STRMS.

Jak widać system ten jest istotnie bardzo elastyczny i dołączanie nowych kanałów nie jest trudne. Problem jedynie w tym, że instrukcje OPEN i CLOSE działają jedynie ze standardowymi identyfikatorami kanałów "K", "S" i "P". Przyłączanie i odłączanie strumieni do nowych kanałów (modyfikacja danych w STRMS oraz zmiennej CURCHL) musi być więc wykonane przez program. Na ogół wymaga to przesłania dodatkowych sygnałów inicjujących dane urządzenie bądź informujących go o zakończeniu sesji.

Na koniec przykra wiadomość dla posiadaczy ZX Interface 1: po przyłączeniu tego urządzenia do Spectrum zmieniają się formaty przechowywania danych o kanałach i strumieniach i powyższe uwagi nie dają się bezpośrednio zastosować.

Rozdział VIII

PROCEDURY SYSTEMOWE

Dla programujących w asemblerze Z80 ROM ZX Spectrum jest cenną składnicą gotowych i przetestowanych procedur w kodzie maszynowym. Kilka z nich może być stosowanych z powodzeniem z poziomu języka Basic stwarzając dodatkowe możliwości niedostępne innymi metodami. Szczupłość miejsca nie pozwala na pełny przegląd pamięci ROM. Ograniczymy się zatem do przedstawienia najważniejszych, naszym zdaniem, procedur.

Przystępując do pisania jakiegokolwiek programu w asemblerze programista musi zazwyczaj umożliwić swemu programowi komunikację z użytkownikiem. Potrzebne są podprogramy do czytania klawiatury, piszące na ekranie, rysujące, drukujące na drukarce, obsługujące magnetofon i głośnik. Drugą grupę problemów stanowi konieczność opracowania procedur zdolnych do przeprowadzenia mniej lub bardziej skomplikowanych obliczeń numerycznych. W ZX Spectrum wszystkie takie podprogramy już są i wystarczy znać ich adresy startowe oraz sposób dostarczania im danych.

Obsługa głośnika

W ZX Spectrum wzbudzać głośnik możemy na dwa różne sposoby:

BEEPER #03B5 = 949

Procedura ta wymaga dwóch parametrów. W rejestrach DE umieszcza się czas trwania dźwięku, a w HL częstotliwość. Odpowiednie wartości, przed umieszczeniem ich w rejestrach, wymagają wstępnych przeliczeń. Przypuśćmy, że chcemy uzyskać ton o częstotliwości f w czasie t. Do DE ładujemy wtedy f*t a do HL 437500/f — 30.125.

Oszczędzić sobie tych rachunków można przez wywołanie procedury

BEEP #03F8 = 1016

Wymaga ona również podania czasu i częstotliwości dźwięku, ale w normalnych jednostkach. Dane te przekazuje się procedurze BEEP przez umieszczenie ich na stosie kalkulatora. BEEP sama je stamtąd zdejmuje, dokonuje niezbędnych przeliczeń i woła BEEPER. Ta ostatnia jest nieco wygodniejsza z tego względu, że nie stosują się do niej ograniczenia nakładane na parametry instrukcji Basic — BEEP. Ich kontrola jest bowiem przeprowadzana w czasie wstępnych przeliczeń wykonywanych przez BEEP. W czasie generowania dźwięku są wyłączane przerwania maskowalne. Ostatnią instrukcją BEEPER przed wykonaniem RET jest EI.

Współpraca z magnetofonem

Zarówno nagłówki jak i bloki danych są nagrywane na kasetę przez tę samą procedurę

SAVE BYTES #04C2 = 1218

W rejestrach DE, w chwili wywołania powinna znajdować się długość nagrywanego bloku, w IX — adres pierwszego bajtu, a w akumulatorze A — typ nagrywanego bloku. System używa dwóch typów: 0 oznacza nagłówek, a 255 = #FF sygnalizuje właściwy blok danych. W zasadzie różnica sprowadza się do tego, że nagłówek jest poprzedzany dłuższym sygnałem wstępnym (ok. 5 sekund) niż właściwy blok (ok. 2 sek.). Użytkownik może wykorzystywać również pozostałe wartości między 0 i 255 na oznaczanie typu bloku. Jest to wygodne przy oznaczaniu różnych typów zbiorów w specjalnych zastosowaniach. Takie bloki są w trakcie czytania instrukcją LOAD... ignorowane (nie są na ekranie drukowane żadne dane o nich), a część prostszych programów kopiujących nie potrafi sobie z nimi poradzić.

Standardowy nagłówek (rozpoznawany przez system) zbudowany jest zawsze z siedemnastu bajtów. Pierwszy bajt zawiera typ bloku danych opisywanych przez konkretny nagłówek. Może to być:

0	program Basic
1	tablica numeryczna
2	tablica znakowa
3	zbiór bajtów

Następnych dziesięć bajtów przeznaczonych jest na kody kolejnych symboli nazwy. Dopuszczalne są wszystkie kody od 0 do 255. Krótsze nazwy system uzupełnia spacjami. Bajty dwunasty i trzynasty zawierają długość bloku następującego po nagłówku. Znaczenie pozostałych czterech bajtów zależne jest od typu zbioru. Dla typu 0 bajty czternasty i piętnasty zawierają numer linii, od której wczytany program ma się automatycznie uruchomić.

Rezygnacja z autostartu sygnalizowana jest wartością większą niż 32767 = #7FFF. Ostatnie dwa bajty zawierają długość samego programu bez obszaru zmiennych. W przypadku tablic jedynie bajt piętnasty jest wykorzystywany i zawiera jednoliterową nazwę nagrywanej tablicy (tę, pod którą tablica była przechowywana w obszarze zmiennych). Dla typu 3 bajty dwunasty i trzynasty zawierają, tak jak dla pozostałych typów, wartość określającą długość bloku. Bajty czternasty i piętnasty określają adres komórki pamięci, w której znajdował się pierwszy element zbioru w momencie nagrania. Tam też zostanie potem wczytany, jeśli parametry LOAD... nie zadecydują inaczej. Dla typów 1, 2, 3 bajty szesnasty i siedemnasty nie mają znaczenia. Powyższy format nagłówka musi być zachowany jedynie wtedy, gdy nagrywane bloki mają być wczytywane instrukcją LOAD... . Wczytywanie z kasety nagranych zbiorów odbywa się za pośrednictwem procedury

LOAD BYTES #0556 = 1366.

Podobnie jak w SAVE BYTES przed wywołaniem, w parze rejestrów DE podajemy długość wczytywanego bloku, w IX — adres pierwszego bajtu, począwszy od którego nowy blok będzie ładowany do pamięci i w akumulatorze A — typ wczytywanego zbioru, czyli 0 lub 255 (lub to, co sami umieściliśmy nagrywając). Dodatkowo jeszcze trzeba instrukcją SCF ustawić znacznik C. Wywołanie tej procedury z wyzerowanym znacznikiem C będzie pozwalało jedynie na dokonanie weryfikacji danego bloku (odpowiednik rozkazu VERIFY). Ewentualny błąd ładowania lub weryfikacji jest sygnalizowany wyzerowaniem znacznika C po wyjściu z podprogramu (prawidłowy przebieg jest sygnalizowany ustawieniem C na 1).

Wyświetlanie i drukowanie

Wyświetlanie pojedynczego symbolu o kodzie zawartym w akumulatorze na górnej i dolnej części ekranu oraz drukowanie na drukarce wykonywane jest przez tę samą procedurę. Przed jej wywołaniem trzeba otworzyć odpowiedni kanał za pomocą procedury

CHAN OPEN #1601 = 5633

Wołamy ją podając w akumulatorze, o który kanał nam chodzi: 1 dla "K", 2 dla "S" i 3 dla "P". Kanał ten będzie otwarty, dopóki go sami nie zamkniemy. Począwszy od tego momentu rozkaz asemblera

RST #10

będzie drukował pojedynczy symbol z A poprzez wybrany kanał.

Poniższy program ilustruje użycie RST # 10. Jego działanie jest równoważne rozkazowi PRINT FLASH 1;AT 5,3;"X";#3;"A"

```
LD    A,2          ;otwarcie
CALL  CHAN_OPEN    ;kanału "S"
LD    A,#12        ;kod symbolu kontrolnego FLASH
RST   #10
LD    A,1          ;argument FLASH
RST   #10
LD    A,#16        ;symbol kontrolny AT
RST   #10
LD    A,5          ;argumenty AT
RST   #10
LD    A,3
RST   #10
LD    A,#58        ;kod "X"
RST   #10
LD    A,3          ;otwarcie
CALL  CHAN_OPEN    ;kanału "P"
LD    A,#41        ;kod "A"
RST   #10
RET                ;wyjście z podprogramu.
```

Zauważmy, że w ostatnim fragmencie znak "A" będzie przesłany do bufora drukarki i faktycznie wydrukowany na papierze dopiero po zapełnieniu bufora, przesłaniu do niego symbolu końca linii (13) lub wywołaniu

COPY_BUFF #0ECD = 3789

Przykład ten ukazuje, że przesyłanie łańcuchów znaków po jednym symbolu może być skrajnie kłopotliwe. Prostsze jest zastosowanie procedury

PR_STRING #203C = 8252

Drukuje ona łańcuch znaków umieszczony w pamięci pod adresem podanym w DE i o długości podanej w BC. Przed jej wywołaniem trzeba oczywiście otworzyć odpowiedni kanał. Jeżeli drukowane ciągi symboli nie precyzują kolorów, to są one ustalane na podstawie zmiennych ATTR_T, MASK_T oraz nieparzystych bitów P FLAG.

Drukowanie liczb jest znacznie bardziej skomplikowane, gdyż niezbędna jest konwersja postaci binarnej liczby na ciąg cyfr jej przedstawienia dziesiętnego. Wszystkie niezbędne przeliczenia i druk wykonuje procedura

PRINT_FP #2DE3 = 11747

Pobiera ona ze stosu kalkulatora pięć bajtów traktując je jako liczbę w postaci przyjętej w systemie Basic. Następnie drukuje ją

uwzględniając odpowiednie zmienne systemowe określające kolory, położenie itp. Sposoby umieszczania liczb na stosie kalkulatora omawiamy dalej.

W przypadku niewielkich liczb naturalnych od 0 do 9999 można użyć nieco szybszej procedury

OUT_NUM1 #1A1B = 6683

Drukuje ona liczbę zawartą w BC na czterech polach uzupełniając z przodu niezbędną liczbę spacji, co czasem może być wygodne. Przypadkowe wywołanie OUT_NUM1 z wartością w BC większą od 9999 nie załamie programu, ale wydruk nie będzie miał wiele wspólnego z drukowaną liczbą. ZX Spectrum wykorzystuje tę procedurę do drukowania numerów linii w listingach programów.

Rysowanie na ekranie

Do rysowania na ekranie mamy odpowiedniki rozkazów PLOT, DRAW i CIRCLE.

PLOT_SUB #22E5 = 8933

Procedura ta pozwala na narysowanie na ekranie pojedynczego punktu o współrzędnych (x,y). Przed jej wywołaniem wystarczy umieścić x w rejestrze C i y w B.

Przy okazji warto wspomnieć o procedurze

PIXEL_ADD #22AA = 8874

Po umieszczeniu w BC wartości y,x i jej wywołaniu otrzymamy w rejestrach HL adres bajtu opisującego dany punkt ekranu, w akumulatorze natomiast umieszczana jest wartość x mod 8 określająca, o który bit tego bajtu chodzi.

Rozkaz DRAW x,y możemy na poziomie asemblera zrealizować na co najmniej dwa sposoby:

DRAW—1 #2477 = 9335

Procedura ta zdejmuje ze stosu kalkulatora liczby x i y, po czym rysuje odpowiedni odcinek. Współrzędne PLOT pobierane są z odpowiednich zmiennych systemowych. Drugi sposób pozwala ominąć operacje ze stosem:

DRAW—3 #24BA = 9402

Do narysowania analogicznego odcinka tą procedurą niezbędne są następujące dane: ABS y w B, ABS x w C, SGN y w D oraz SGN x w E.

Fragmenty łuków rysuje

DRAW_ARC #2394 = 9108

Parametry x, y, z dla tej procedury muszą być przekazane przez stos kalkulatora. Na szczycie stosu lokujemy z.

Pełny okrąg o środku x,y i promieniu z rysuje

CIRCLE_1 #232D = 9005

Tu również wszystkie parametry muszą być umieszczone na stosie kalkulatora.

UWAGA: Powyższe procedury rysujące modyfikują rejestry H'L' (patrz "Błędy w systemie" o problemach z RET).

Po wykonaniu efektownego rysunku możemy wydrukować go na drukarce. W pamięci ROM odpowiednikiem instrukcji COPY jest instrukcja

COPY #0EAC = 3756

Procedura ta kopiuje na drukarce 22 górne linie ekranu. Parametry wszystkich powyższych procedur muszą spełniać te same ograniczenia, co odpowiednie rozkazy w języku Basic.

Czyszczenie i przesuwanie ekranu

Podstawową procedurą do czyszczenia ekranu jest

CLS #0D6B = 3435

Działanie jej jest identyczne z rozkazem o tej samej nazwie. Dolną część ekranu można czyścić procedurą

CLS_LOWER #0D6E = 3438

Działa ona poprawnie na całej dolnej części ekranu niezależnie od jej chwilowych rozmiarów i równocześnie ustala jej wysokość na dwie linie. Inicjowane są także zmienne systemowe: DF_CL i SPOSNL, określające położenie kursora.

Dodatkową możliwością jest czyszczenie z góry zadanej liczby wierszy, licząc od dołu ekranu. Ten efekt można uzyskać przy pomocy procedury

CL_LINE #0E44 = 3652

Liczbę linii do wymazania podajemy w rejestrze B.

Przed wywołaniem tych procedur trzeba się upewnić, że są otwarte odpowiednie kanały. Pierwsze dwie z nich na wyjściu pozostawiają otwarty kanał "K". Przy wymazywaniu ekranu kolory są ustalane na podstawie zmiennych systemowych BORDCR dla dolnej części ekranu oraz ATTR_P i MASK_P dla górnej, przy czym ich zawartość jest kopiowana do ATTR_T i MASK_T.

Zawartość całego ekranu (pełne 24 wiersze) można przesuwać o jeden wiersz do góry (znika wtedy ten na samej górze) procedurą

CL_SC_ALL #0DFE = 3582

Można ją wywoływać bezpośrednio z ZX—Basic, gdyż nie wymaga żadnych parametrów.

Ciekawsza od niej jest procedura

CL_SCROLL #0E00 = 3584

Po podaniu w rejestrze B liczby linii do przesunięcia minus 1 (ale nie mniej niż dwie) procedura ta przesunie o jeden wiersz tyle linii, ile chcemy, nie ruszając leżących powyżej. Można więc mieć na górze ekranu rysunek lub tekst i nie niszcząc go przesuwać o jeden wiersz teksty pojawiające się poniżej. Przy stosowaniu tych procedur trzeba pamiętać, że przesuwane go góry będą również atrybuty obowiązujące w dolnej części ekranu.

Czytanie klawiatury

Informację z klawiaturu najwygodniej jest pobierać ze zmiennej systemowej LAST_K. Badając piąty bit zmiennej FLAGS sprawdzamy, czy został wciśnięty kolejny klawisz, czy jeszcze nie (1 — wciśnięto nowy, 0 — jeszcze nie wciśnięto żadnego od czasu wyzerowania tego bitu). Po wczytaniu kodu klawisza zerując ten bit zapewniamy sobie możliwość odróżnienia, czy zawartość LAST_K została już zmodyfikowana, czy jeszcze nie.

System ten działa jedynie przy włączonych przerwaniach maskowalnych w trybie 1. Wtedy to bowiem Spectrum automatycznie wywołuje 50 razy w ciągu sekundy procedurę.

KEYBOARD #02BF = 703

Przegląda ona całą klawiaturę identyfikując poprawną kombinację klawiszy, umieszcza wczytany kod w akumulatorze oraz zmiennej LAST_K i ustawia piąty bit FLAGS.

Interpretacja wciśniętego klawisza zależna będzie od trzech zmiennych systemowych. Jako pierwsza testowana jest zawartość MODE traktowana jako liczba ze znakiem w kodzie uzupełnień do dwóch:

MODE — 1<0 oznacza kursor K, L lub C,
MODE — 1=0 oznacza kursor E,
MODE — 1>0 oznacza kursor G.

Kursor K od L i C odróżniany jest za pośrednictwem trzeciego bitu zmiennej FLAGS. Zero oznacza K, a jeden sygnalizuje L lub C. Trzeci bit zmiennej FLAGS2 ostatecznie rozstrzyga, czy kursorem jest L (0), czy C (1). Procedura ta nie czeka na wciśnięcie klawisza.

Aby niezależnie od stanu kursora stwierdzić, czy został wciśnięty konkretny klawisz wygodniejsze może być bezpośrednie badanie klawiatury za pomocą instrukcji IN. Postąpimy tu identycznie, jak w przypadku funkcji o tej samej nazwie w języku Basic.

Kalkulator

Różne operacje na liczbach zmiennoprzecinkowych są wykonywane przez duży zestaw procedur zajmujących obszar od #2F9B = 12187 do #386D = 14445. **Kalkulator wywołuje się instrukcją RST #28, która wykonuje skok pod adres #335B = 13147**

Podstawą działania kalkulatora jest korzystanie z 66 różnych procedur wykonujących zestaw podstawowych operacji na stosie kalkulatora. O kolejności ich wywołania decyduje ciąg bajtów umieszczany bezpośrednio za rozkazem RST #28. Koniec takiego ciągu zawsze jest sygnalizowany bajtem o wartości #38 = 56.

Z konieczności ograniczymy się do operacji pozwalających wykonywać wszelkie obliczenia numeryczne. Załóżmy, że na szczycie stosu znajdują się liczby ... z x y.

wartość bajtu dec	hex	**operacja**	**stan stosu po operacji**			
1	#01	zamiana elementów na szczycie		...z	y	x
3	#03	odejmowanie			... z	x—y
4	#04	mnożenie			... z	x*y
5	#05	dzielenie			... z	x/y
6	#06	potęgowanie			... z	x^y
15	#0F	dodawanie			... z	x+y
27	#1B	negowanie		... z	x	—y
31	#1F	sinus		... z	x	sin y
32	#20	cosinus		... z	x	cos y
33	#21	tangens		... z	x	tan y
34	#22	arcus sinus		... z	x	asn y
35	#23	arcus cosinus		... z	x	acs y
36	#24	arcus tangens		... z	x	atn y
37	#25	logarytm naturalny		... z	x	ln y
38	#26	funkcja wykładnicza ex		... z	x	exp y
39	#27	część całkowita liczby		... z	x	int y
40	#28	pierwiastek kwadratowy		... z	x	sqr y
41	#29	znak liczbowy		... z	x	sgn y
42	#2A	wartość bezwzględna liczby		... z	x	abs y
49	#31	kopiowanie szczytu stosu	... z	x	y	y
50	#32	n mod m dzielenie z resztą		... z	reszta	iloraz
52	#34	dopisanie danych na stos	... z	x	y	d
56	#38	koniec obliczeń		... z	x	y
88	#3A	INT(y+.5)		... z	x	INT(y+.5)
160	#A0	dopisanie zera na stos	... z	x	y	0
161	#A1	dopisanie jedynki na stos	... z	x	y	1
162	#A2	dopisanie 1/2 na stos	... z	x	y	1/2
163	#A3	dopisanie PI/2 na stos	... z	x	y	PI/2
164	#A4	dopisanie 10 na stos	... z	x	y	10

Jako przykład użycia kalkulatora obliczmy wartość 1.3*sinx + x ↑ 2*cos(x PI/2). Załóżmy, że wartość x już znajduje się na szczycie stosu. Umieszczenie za instrukcją RST #28 kolejnych bajtów z pierwszej kolumny spowoduje

bajt	stan stosu po operacji				
#31	x	x			
#31	x	x	x		
#31	x	x	x	x	
#A3	x	x	x	x	PI/2
#04	x	x	x	x*PI/2	
#20	x	x	x	cos(x*PI/2)	
#04	x	x	x*cos(x*PI/2)		
#04	x	x*x*cos(x*PI/2)			
#01	x*x*cos(x*PI/2)	x			
#1F	x*x*cos(x*PI/2)	sin x			
#34	polecenie dopisania na szczyt stosu podanej w dalszych				
#F1	5 bajtach stałej — w tym wypadku 1.3				
#26					
#66					
#66					
#66	x*x*cos(x*PI/2)	sin x	1.3		
#04	x*x*cos(x*PI/2)	1.3*sin x			
#0F	x*x*cos(x*PI/2)+1.3*sin x				
#38	koniec obliczeń				

Wyjaśnienia wymaga sposób przekazywania na stos akumulatora danych jako ciągu bajtów następujących bezpośrednio za #34. Uważny Czytelnik z pewnością spostrzegł, że bajty #F1 #26 #66 #66 #66 przedstawiają liczbę 2 ↑ 113*0.13, a nie 1.3. Wiąże się to z interpretacją ich jako liczby w zapisie skróconym. Schemat postępowania jest następujący:

i/ pierwszy bajt dzielimy przez #40 = 64 i jako wartość wykładnika przyjmuje się
— resztę z tego dzielenia plus #50, jeśli jest ona różna od zera, lub
— drugi z kolei bajt plus #50, jeśli reszta ta równa jest 0,
ii/ iloraz wykonanego dzielenia (0,1,2,3) plus 1 określa, ile podano bajtów mantysy. Brakujące bajty (do pięciu) uzupełnia się zerami.

W naszym przykładzie #F1 = 241 dzielone przez #40 = 64 daje resztę #31 = 49 oraz iloraz 3. Oznacza to, że wykładnikiem naszej liczby jest #31 + #50 = #81 i że podano wszystkie cztery bajty mantysy.

W systemie tym liczba 0 ma przedstawienie #40 #B0 #00, bo 0 dzielone przez #40 daje resztę i iloraz równy zeru. Zatem wykładnikiem jest drugi bajt plus #50, czyli #B0 + #50 = 0 (wszystkie te obliczenia na pojedynczych bajtach prowadzi się modulo 256) i podany jest tylko pierwszy bajt mantysy. Pozostałe bajty (do pięciu) uzupełniamy zerami.

Z kolei liczba 10 ma przedstawienie #40 #B0 #00 #0A.

Symbol #34 pozwala na umieszczanie danych liczb w tekście programu asemblerowego. Często jednak trzeba umieścić na stosie parametry instrukcji czy wartości obliczane w programie. Służy temu wiele procedur pomocniczych. Umożliwiają one zarówno zdejmowanie ze stosu wartości, jak i umieszczanie na nim różnych liczb. Zaliczymy do nich

STK_TO_BC #2307 = 8967

Dwie kolejne liczby (w postaci 5—bajtowej) są zdejmowane ze stosu i umieszczane w rejestrach B i C. Ich wartości nie powinny przekraczać przedziału —255...255, bo nastąpi powrót do systemu Basic z komunikatem B (wykonana zostanie instrukcja RST # 8). Procedura

ta może być stosowana do liczb ujemnych. Znaki ich są zwracane w rejestrach D i E. Liczby zdejmowane ze stosu są oczywiście przybliżane do najbliższej liczby całkowitej.

STK_TO_A #2314 = 8980

Procedura analogiczna do poprzedniej z tym, że ze stosu zdejmowana jest tylko jedna liczba i po przybliżeniu do wartości całkowitej umieszcza się ją w akumulatorze. Jej znak wraca w rejestrze C.

STACK_FETCH #2BF1 = 11249

Ta procedura zdejmuje ze stosu całą liczbę (pięć bajtów) i rozmieszcza je kolejno w rejestrach A,E,D,C,B.

FP_TO_BC #2DA2 = 11682

Liczba ze stosu zostaje przybliżona przez najbliższą liczbę całkowitą i umieszczona w rejestrach BC. Znak liczby sygnalizowany jest znacznikiem Z (0 dla liczb ujemnych). Jeśli wartość na stosie przekraczała co do wartości bezwzględnej 65535 znacznik C ustawiany jest na 1 i jest to jedyna reakcja na błąd (nie następuje powrót do ZX—Basic z odpowiednim komunikatem).

Z kolei do umieszczania wartości na stosie może służyć

STK_STORE #2AB6 = 10934

Procedura ta lokuje na szczycie stosu pięć bajtów z kolejnych rejestrów A,E,D,C,B. Liczbę 1.3 (#81 #26 #66 #66 #66) można umieścić na stosie kalkulatora na dwa sposoby:

```
LD    A, #81
LD    DE, #6626
LD    BC, #6666
CALL  #2AB6
lub
RST   #28
DEFB  #34
DEFB  #F1
DEFB  #26
DEFB  #66
DEFB  #66
DEFB  #66
DEFB  #38
```

Zauważmy. że drugi sposób pozwala zaoszczędzić trzy bajty.

STACK_A #2D28 = 11560

Wartość bezwzględna rejestru A jest umieszczona na szczycie stosu w postaci 5—bajtowej.

STACK_BC #2D2B = 11563

Procedura analogiczna do poprzedniej, ale na stos odkłada liczbę z pary rejestrów BC.

Po zakończeniu obliczeń kalkulator w rejestrach HL umieszcza adres pierwszego z pięciu bajtów leżących na szczycie stosu. Przy pracy z kalkulatorem trzeba dbać o właściwą obsługę stosu. Pamiętać też trzeba, że procedura PRINT_FP zdejmuje ze stosu drukowaną liczbę.

Na koniec podajmy sposób, jak z programu w kodzie maszynowym można wywoływać generator pseudolosowy. Poniższy program

```
LD    A, #A5
CALL  STACK_A
RST   #28
DEFB  #2F
DEFB  #1D
DEFB  #38
RET
```

umieszcza kolejną liczbę pseudolosową na stosie kalkulatora oraz modyfikuje zmienną systemową SEED.

Inne przydatne procedury

Globalne czyszczenie obszarów roboczych systemu Basic, w tym również stosu kalkulatora, wykonuje procedura

SET_MIN #16B0 = 5808

W rzeczywistości modyfikuje ona zmienne systemowe wskazujące na odpowiednie obszary. Fizycznie zniszczone są tylko te bajty, w które wpisuje się znaczniki końca obszaru.

W programach operujących tekstami w języku Basic przydatna może być procedura

MAKE_ROOM #1655 = 5717

Wywołując ją należy w rejestrach HL podać adres bajtu, w którym rozpoczynać się będzie dodany blok, a w BC rozmiar bloku. Procedura ta sama sprawdza, które zmienne systemowe mają być zmodyfikowane i w razie potrzeby to robi. Pozwala więc ona na wstawianie w istniejący program Basic lub w obszar zmiennych nowych linii, zmiennych itp.

Odwrotną funkcję do powyższej procedury pełni

RECLAIM—2 #19E8 = 8168

Tu z kolei w HL podajemy adres pierwszego bajtu, który ma być usunięty, a w BC rozmiar wyrzucanego bloku. Podobnie jak poprzednio odpowiednie zmienne systemowe zostaną automatycznie zmodyfikowane.

Do procedur porządkowych zaliczymy również

CLEAR_BUFF #0EE7 = 3815

Czyści ona bufor drukarki i modyfikuje związane z nim zmienne systemowe.

Odszukanie adresu linii o danym numerze w programie Basic umożliwia procedura

LINE_ADDR #196E = 6510

Przed wywołaniem w HL podajemy numer poszukiwanej linii. Na wyjściu w HL mamy adres tej linii lub pierwszej o numerze większym. W każdym przypadku w DE jest podany adres poprzedniej linii. Jeśli linia o zadanym numerze istnieje w programie, to jej zlokalizowanie jest sygnalizowane ustawieniem znacznika Z.

Oszacowanie wolnej pamięci w obszarze Basic, czyli między STKEND i RAMTOP daje procedura

FREE_MEM #1F1A = 7962

Rejestry HL i BC zawierają tę samą liczbę ujemną przedstawioną w kodzie uzupełnień do dwóch, będącą różnicą STKEND + 80 oraz SP (wskaźnika stosu procesora Z80).

Na koniec wspomnijmy o procedurze

BREAK_KEY #1F54 = 8020

Wywołuje się ją celem sprawdzenia, czy są jednocześnie wciśnięte klawisze CS i BREAK. Jeśli tak, to znacznik C jest wyzerowany.

Rozdział IX

BŁĘDY W SYSTEMIE

Autorzy systemu operacyjnego i interpretera Basic w ZX Spectrum wykonali wspaniałą robotę. Nie zdołali jednak ustrzec się przed kilkoma błędami. Niektóre z nich są mało istotne, inne zaś nieświadomego programistę mogą doprowadzić do ciężkiej frustracji. Poniżej, ku przestrodze wszystkim użytkownikom, przedstawiamy listę znanych nam błędów zauważonych w tym świetnym skądinąd mikrokomputerze.

1. Błąd dzielenia. Pod adresem #3200 umieszczono wartość #E1 zamiast #DA. W konsekwencji gubiony bywa ostatni bit, co prowadzi do niewłaściwych zaokrągleń. Skutki błędu demonstruje poniższy program:

```
10 LET a=a/b
20 IF a THEN GO TO 10
30 PRINT "Osiągnięto zero"
```

Uruchamiając go z wartościami a=1 i b=3, po sekundzie uzyskamy wydruk "Osiągnięto zero". Uruchamiając go jednak ponownie, tym razem z wartościami a=1 i b=2, wpadamy w pętlę nieskończoną bo Spectrum sądzi, że 2 ↑ —128 = 2 ↑ —128/2.

2. Błąd "—65536". Autorzy dopuścili się niekonsekwencji w przedstawieniu tej liczby. Raz jest ona przechowywana w postaci zmiennoprzecinkowej, a niekiedy jako liczba całkowita w kodzie uzupełnień do dwóch. Skutki niejednoznaczności objawiają się np. w następującym rozkazie PRINT INT —65536. Na ekranie pojawia się liczba ...—1.

3. Błąd CHR$ 8. Wydruk tego symbolu kontrolnego powinien przesunąć kursor na ekranie o jedną pozycję w lewo lub na koniec poprzedniej linii. Tak istotnie jest w liniach 1...23. Na skutek błędu nie można jednak przejść z początku pierwszej linii na koniec tej o numerze 0. Do "ciekawych" efektów prowadzi prośba o przesuwanie kursora w lewo od pola (0,0). Radzimy to sprawdzić, ale raczej bez ważnych danych w pamięci komputera.

4. CHR$ 9. Ten symbol powinien dla odmiany przesuwać kursor o jedno pole w prawo. Tu popełniono jednak poważniejszy błąd. Wyświetlając go nic nie wskóramy. Wykonane są bowiem wszystkie niezbędne obliczenia, ale zapomniano zmodyfikować zmienne systemowe.

5. Błąd "Press any key...". Co najmniej w kilku sytuacjach Spectrum przerywa pracę i czeka na ponaglenie poprzez wciśnięcie któregokolwiek klawisza przez użytkownika. Błąd sprowadza się do tego, że komputer nie reaguje na klawisz CS ani SS, choć reaguje na oba razem.

6. Błąd wskaźnika linii bieżącej. Przypuśćmy, że ostatnia linia w programie ma np. numer 1000. Po wciśnięciu 1001 i ENTER, a następnie CS/1 na dół ekranu zostanie skopiowana linia 1000, ale razem ze wskaźnikiem linii bieżącej, który przed odesłaniem linii trzeba osobno kasować.

7. Błąd DELETE. Przy kasowaniu zawartości dolnego ekranu przez CS/1 zostaje na dół skopiowana linia bieżąca programu i nie jest przywracany właściwy rozmiar tego obszaru. W praktyce trzeba ponownie wciskać ENTER.

8. Błąd wiodących spacji. Niektóre słowa kluczowe są podczas wyświetlania poprzedzane spacją, nie zawsze jednak: spróbujcie np. wykonać PRINT CHR$ 255; CHR$ 13; CHR$ 255.

9. Błąd trybu K. Po wciśnięciu klawisza w trybie K i jego przytrzymaniu, klawisz zaczyna się powielać. Kursor się zmienia na L lub C, ale drukowany jest cały czas symbol w trybie K.

10. Błąd SCREEN$. W komórce #257D winno być #C9 zamiast #C3. W efekcie program

```
10 PRINT "1234567890"
20 LET a$=SCREEN$ (0,0) + SCREEN$ (0,1)
30 PRINT a$
```

zamiast spodziewanej liczby 12 wyświetli 22. Jeszcze dziwniejsze wydruki otrzymamy, gdy na końcu linii 20 dopiszemy + SCREEN$ (0,2) + SCREEN$ (0,4). Zmienna przyjmie wartość "55"!

Ten błąd łatwo ominąć dodając do a$ wartości SCREEN$ (0,i) pojedynczo, a nie w jednym wyrażeniu.

11. Błąd STR$. Działając z liczbami —1<x<1, ale różnymi od zera, można się "nadziać" na niezwykłą "żarłoczność" rozkazu STR$. Spróbujcie wykonać poniższe rozkazy:

```
PRINT "ALA" + "Bum cyk cyk" + STR$ .001
PRINT 7 + VAL STR$ .001
```

W obu przypadkach Waszym zdumionym oczom na ekranie ukaże się jedynie .001. Winę ponosi Spectrum!

12. Błąd CLOSE. Próby odłączenia strumienia 4...15 od kanału przed uprzednim jego przyłączeniem prowadzi do nieprzewidzianych efektów z restartem systemu włącznie. A wszystko dlatego, że w ROM w tablicy zawierającej dane o kanałach pod adresem #1716 zapomniano umieścić znacznik końca tablicy.

13. Błąd RET. Jeden z trudniejszych do zlokalizowania błędów. Objawia się czasami przy powrocie do ZX—Basic z programu w kodzie maszynowym uruchamianego przez USR k. Zapomniano bowiem, przed powrotem do interpretera, odtworzyć wartość pary rejestrów H'L'. Jeśli Wasz program ich używał i zmodyfikował, to to, co się stanie przy powrocie do ZX—Basic jest kwestią przypadku. Zazwyczaj następuje zawieszenie się systemu, ale nie zawsze. Co najciekawsze, wartość ta jest zawsze stała i powinna być równa #2758 = 10072.

14. Błąd NMI. Jest to jeden z najpoważniejszych błędów. Przestawienie jednego bitu pod adresem # 006D sprawiło, że Spectrum nie jest w stanie przyjmować tzw. przerwań niemaskowalnych z obsługą przerwania przez procedurę użytkownika. Przerwania takie mogą być bądź ignorowane, bądź restartować system przez skok do adresu 0. Uniemożliwia to kontrolowany restart systemu po jego załamaniu, jak i wiele innych zastosowań komputera. Eliminacja tego błędu wymaga niełatwych i kosztownych przeróbek technicznych.

15. PAUSE n — błąd. Problem z instrukcją PAUSE polega na tym, że ona nie zawsze działa. Błąd w procedurze obsługującej samopowtarzalność klawiszy sprawia, że jeśli jakiś klawisz był naciskany tuż przed wykonaniem instrukcji PAUSE, to jest ona ignorowana. Zilustrujmy to przykładem:

```
10 PRINT "Puść klawisz gdy usłyszysz dźwięk"
20 FOR i=1 TO 500 ; NEXT i
30 BEEP 1,10
40 PAUSE 0
50 PRINT "Koniec"
```

Zachowanie się programu niczym nie wskazuje na istnienie w nim linii 40.

16. CLS — błąd. Przy wartościach zmiennej systemowej DF SZ mniejszych od 2 trudno uznać efekt tego rozkazu za czyszczenie ekranu. Dla DF SZ = 1 można to uznać za ciekawostkę, ale przy DF SZ = 0 to już tragedia! Sprawdźcie!

Mamy nadzieję, że powyższa lista wyczerpuje "niespodzianki", na które można się natknąć programując ZX Spectrum.

Miłych i owocnych godzin przy klawiaturze życzy Wam

Autor
wraz z zespołem
Popularnego Miesięcznika Informatycznego
KOMPUTER

ABC „KOMPUTEREA"
Andrzej Kadlof
Tajniki ZX Spectrum
seria redagowana przez zespół miesięcznika „KOMPUTER":
Władysław Majewski (redaktor serii),
Elżbieta Bobrowska (sekretarz serii),
Grzegorz Czapkiewicz (recenzent tomu),
Magdalena Stachorzyńska (skład komputerowy),
Stefan Szczypka (opracowanie graficzne),
Małgorzata Luzińska (redaktor techniczny),
wydawca: Krajowe Wydawnictwo Czasopism RSW „Prasa—Książka—Ruch",
00—666 Warszawa, ul. Noakowskiego 14
druk: Prasowe Zakłady Graficzne RSW „P—K—R"
Łódź, ul. Armii Czerwonej 28. Zam. 3700/86
nakład: 80 tys. egz.
cena: 100 zł K-82

Już wkrótce w tej samej serii:

Roland Wacławek "ABC Commodore 64", cz.1 i 2.

Wojciech Wojtanowski "Z Amstradem (Schneiderem) za pan brat"

ABC Amstrada (Schneidera) 464, 664, 6128

Zbigniew Kasprzycki "LOGO"

Jarosław Kania "LOGO — przykłady programów"

Cena 100 zł